Uitdagingen en oplossingen in H.265/HEVC
voor integratie van consumentenelektronica in professionele videosystemen

Challenges and Solutions in H.265/HEVC
for Integrating Consumer Electronics in Professional Video Systems

Thijs Vermeir

Promotoren: prof. dr. P. Lambert, dr. ir. J. Slowack
Proefschrift ingediend tot het behalen van de graad van
Doctor in de industriële wetenschappen

UNIVERSITEIT
GENT

Vakgroep Elektronica en Informatiesystemen
Voorzitter: prof. dr. ir. R. Van de Walle
Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2016 - 2017

ISBN 978-90-8578-972-7
NUR 965
Wettelijk depot: D/2017/10.500/7

Promotoren:	dr. ir. Jürgen Slowack (Barco) prof. dr. Peter Lambert (Universiteit Gent)
Examencommissie:	prof. dr. ir. Rik Van de Walle (Universiteit Gent) dr. ir. Glenn Van Wallendael (Universiteit Gent) prof. dr. ir. Filip De Turck (Universiteit Gent) prof. dr. ir. Steven Verstockt (Universiteit Gent) prof. dr. ir. Adrian Munteanu (Vrije Universiteit Brussel) dr. ir. Jean-Francois Macq (Nokia, Antwerpen) dr. Kristof Denolf (Xilinx, San José, Verenigde Staten)
Interne verdediging:	9 december 2016
Publieke verdediging:	24 januari 2017

Universiteit Gent
Faculteit Ingenieurswetenschappen en Architectuur
Vakgroep Elektronica en Informatiesystemen
IDLab

Campus Ufo,
Sint-Pietersnieuwstraat 41,
B-9050 Gent, België

Tel.: +32-9-331.49.93

Dankwoord

Vier jaar geleden ben ik een project gestart dat nu op zijn einde loopt met dit boek. Ik ben trots dat ik dit heb kunnen bereiken, toch is de kans klein dat ik uit dit boek zal voorlezen aan mijn kleinkinderen. De technieken besproken in dit boek zullen met de tijd verbeterd worden en uiteindelijk irrelevant worden. De kennis, ervaringen en herinneringen die ik heb opgedaan zijn daarentegen wel blijvend. Het zou niet mogelijk zijn geweest om op dit punt te komen zonder de hulp van andere mensen, deze zou ik hier willen bedanken.

Eerst en vooral zou ik iedereen willen bedanken die heeft geholpen bij dit project. Van bij het bedenken tot het einde van het project heb ik hulp gehad van heel wat mensen van Barco en Universiteit Gent. In het bijzonder zou ik mijn promotoren, Jürgen Slowack en Peter Lambert, willen bedanken voor hun hulp, geduld en aanmoediging bij het onderzoek. Zonder hun hulp was het niet mogelijk geweest om dit onderzoek uit te voeren en succesvol af te ronden.

Verder wil ik iedereen bedanken voor het grondig nalezen van mijn publicaties en dit proefschrift. In het bijzonder verdienen de leden van mijn examencommissie een extra vermelding aangezien hun opmerkingen en suggesties de kwaliteit van dit werk in grote mate verhoogd hebben.

Ik zou ook Barco en alle Barco collega's die in dit project geloofden en hieraan hebben bijgedragen willen bedanken voor hun steun en hulp.

Daarnaast zou ik ook alle collegas van Multimedia Lab / Data Science Lab / IDLab willen bedanken. Ze zorgden voor een aangename werksfeer en stimulerende omgeving.

Mijn ouders wil ik bedanken omdat ze me altijd gesteund hebben en met raad en daad bijgestaan indien nodig. Zonder hun hulp zou ik zelfs niet op het punt hebben kunnen komen om dit project te starten. Mijn moeder en schoonmoeder zijn er spijtig genoeg niet meer bij, maar ook aan hen heb ik veel te danken. Ik ben zeker dat ze nu trots op me geweest zouden zijn.

Tot slot, en zeker niet het minst, wil ik mijn vrouw Annelore bedanken. Samen werden we in deze periode trotse ouders van onze twee prinsesjes Lily en Fleur. Bedankt voor alle hulp en ondersteuning, jullie hebben een speciale plaats in mijn hart.

Staden, januari 2017
Thijs Vermeir

Table of Contents

List of Figures

List of Tables

List of Acronyms

AMVP	Advanced Motion Vector Prediction
ASIC	Application-Specific Integrated Circuit
AVC	Advanced Video Coding
BARCO	Belgian-American Radio Company
BD-rate	Bjøntegaard delta rate
BYOD	Bring Your Own Device
BYOD	Bring Your Own Device
CABAC	Context-Adaptive Binary Arithmetic Coding
CB	Coding Block
CCE	Complexity Constrained Encoding
COTS	Commercial Off-The-Shelf
CSRC	Contributing Source
CT	Complexity Target
CTU	Coding Tree Unit
CU	Coding Unit
dB	decibel
DCT	Discrete Cosine Transform
DiOR	Digital Operating Room
DPB	Decoder Picture Buffer
DSC	Display Stream Compression
DST	Discrete Sine Transform
ESD	Early Skip Detection
ET	Encoding Time
FPGA	Field-Programmable Gate Array
GCR	Guided Chroma Reconstruction
GPU	Graphics Processing Unit
GW	GateWay
HEVC	High Efficiency Video Coding
HM	HEVC Test Model
HQ	High Quality
IaaS	Infrastructure as a Service
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPU	Image Processing Unit
ISO	International Standards Organization
ITU-T	International Telecommunications Union

kbps	kilobit per second
LAN	Local Area Network
LCD	Liquid Crystal Display
LED	Light-Emitting Diodes
MANE	Media Aware Network Element
Mbps	megabit per second
MD2	Maximum coding unit depth of 2
MDO	Minimum Difference Offset
MOPS	Millions of Operations executed Per Second
MPEG	Moving Picture Experts Group
MPLS	Multiprotocol Label Switching
MSE	Mean Square Error
MTU	Maximum Transmission Unit
MV	Motion Vector
NAL	Network Abstraction Layer
NTP	Network Time Protocol
PDV	Packet Delay Variation
PPS	Picture Parameter Set
PSNR	Peak Signal to Noise Ratio
PT	Payload Type
RAP	Random Access Point
RD	Rate-Distortion
RDC	Rate-Distortion-Complexity
Rext	Format Range Extension
RFB	Remote FrameBuffer
RTCP	RTP Control Protocol
RTP	Real-time Transport Protocol
SAO	Sample Adaptive Offset
SCADA	Supervisory Control and Data Acquisition
SCC	Screen Content Coding
SHT	Super High Tier
SoC	System on Chip
SPS	Sequence Parameter Set
SSD	Sum of Squared Differences
SSIM	Structural similarity
SSRC	Synchronization Source
TID	Temporal Identifier
UHD	Ultra High Definition
USB	Universal Serial Bus
VCEG	Video Coding Experts Group
VNC	Virtual Network Computing
VPS	Video Parameter Set
WPP	Wavefront Parallel Processing

Nederlandse samenvatting –Summary in Dutch–

Videostreamingapplicaties zijn alomtegenwoordig in ons dagelijkse leven, en de ondersteuning voor deze diensten via hardware en software is vanzelfsprekend op de meeste moderne consumentenelektronica zoals laptops, tablets, smartphones en televisies. Deze consumentenelektronica kan echter niet zomaar geïntegreerd worden in professionele videosystemen (bv. voor videobeveiliging, evenementen, onderwijs), omwille van de veel hogere vereisten van deze professionele toepassingen wat betreft betrouwbaarheid, snelheid en beeldkwaliteit. Het doel van dit doctoraatswerk is om te onderzoeken hoe consumentenelektronica en meer bepaald, veelgebruikte videocompressiestandaarden (met in het bijzonder, H.265/HEVC) op een kostenefficiënte manier toch gebruikt kunnen worden in professionele applicaties en dit zonder in te boeten aan kwaliteit.

Consumentenelektronica heeft typisch alleen hardware-ondersteuning voor de meer conventionele videocoderingsprofielen zoals H.265/HEVC Main profile. Een dergelijk profiel ondersteunt alleen video met 4:2:0 chromasubsampling. Voor synthetische / computergegenereerde beelden (zoals presentatieslides, werkbladen, enz.) creëert deze chromasubsampling afwijkingen die storend kunnen zijn voor professionele videoapplicaties. Daarom, is een eerste belangrijke bijdrage in dit onderzoek een techniek voor het herstellen van de afwijkingen geïntroduceerd door chromasubsampling.

In een eerste benadering wordt een aangepaste *guided image filter* gebruikt om de chromacomponent van de gedecodeerde videostroom te verbeteren met behulp van de lumacomponent. Een *guided image filter* met vaste configuratie kan het beeld niet verbeteren voor computergegenereerde beelden. Omdat de verbeteringen van deze filter heel lokaal zijn werd er een meer adaptieve methode onderzocht. Gebaseerd op het histogram van de lumacomponent wordt er een voorspellingsmodel opgesteld. Dit model wordt gebruikt om de radiusparameter van de *guided image filter* in te stellen. Dit werkt goed voor computergegenereerde beelden met een beperkt aantal unieke kleuren. Het model werkt niet goed als er veel meer kleuren zijn (zoals bijvoorbeeld bij een gradient of bij transparante gebieden), er worden zelfs nieuwe fouten geïntroduceerd in dit geval. Dit komt omdat er bij veel kleuren weinig gelijkenis is tussen de optimale radiusparameter van omliggende pixels, wat het moeilijk maakt om een goed voorspellingsmodel te vinden. Daarom werd er overwogen om de videostroom uit te breiden met extra informatie. Hiermee kan de decoder de *guided image filter* per pixel herconfigureren. Deze

methode voldoet aan de videocompressiestandaard (i.e., H.265/HEVC), omdat de extra informatie die toegevoegd wordt aan de videostroom niet zal geïnterpreteerd worden door decoders die deze uitbreiding niet ondersteunen (e.g., decoders van andere producenten). Met het Minimale Verschil-Afwijking (Minimum Difference Offset, MDO) principe, zal de encoder enkel aangeven om de *guided image filter* te gebruiken als er een grote verbetering is in het beeld. Hierdoor worden er geen nieuwe afwijkingen in het beeld geïntroduceerd. Door de MDO te configureren heeft de gebruiker de mogelijkheid om te kiezen tussen beeldkwaliteit en bandbreedte. Een nadeel van deze methode is dat deze een aanpassing nodig heeft aan zowel de encoder als aan de decoder.

Een tweede benadering voor het herstellen van de kleurinformatie is daarom Gestuurde Chroma Reconstructie. Met deze methode is het ook mogelijk om computergegenereerde beelden te herstellen die geëncodeerd zijn door andere encoders (bv. op consumentenelektronica). Deze methode heeft geen extra informatie nodig van de encoder. De methode is net zoals de vorige methode gebaseerd op het feit dat er correlatie is tussen de luma- en chromacomponent. Daarnaast houdt ze ook rekening met de subsamplingfilter die de encoder gebruikt heeft. Een eerste veronderstelling is dat als twee pixels dezelfde lumawaarden hebben, ze dan ook dezelfde chromawaarden hebben. De tweede veronderstelling gaat ervan uit dat de luma- en chromacomponent dezelfde structuur hebben. Gestuurde Chroma Reconstructie zal de pixel enkel trachten te herstellen als er een hoge kans is dat deze veronderstellingen correct waren. Deze methode werkt zowel in het geval van verliesloze als niet-verliesloze compressie. De methode werkt goed voor synthetische beelden en de impact op natuurlijke beelden is minimaal omdat er bij dergelijke beelden weinig correlatie wordt gevonden tussen de luma- en chromacomponent. Als gevolg is het niet nodig om de karakteristieken van de videostroom te detecteren voordat deze methode wordt gebruikt.

Omdat hardwareversnelling niet altijd beschikbaar is op consumentenelektronica, wordt er als een tweede belangrijke bijdrage in dit onderzoek technieken voorgesteld om de snelheid en betrouwbaarheid te verbeteren door complexiteitsbeperkingen op te leggen bij softwaregebaseerde videocompressie. In de huidige softwaregebaseerde implementaties, is de tijd die nodig is om een beeld te encoderen afhankelijk van meerdere factoren zoals de karakteristieken van de beeldinhoud, de mogelijkheden van de hardware en (manuele) parameterconfiguratie. Dit veroorzaakt echter variabiliteit en onvoorspelbaarheid, wat resulteert in beelden die mogelijks wegvallen of hardware die wordt overgedimensioneerd. Door de software-encoder aan te passen zodat deze rekening houdt met een gegeven complexiteitsbudget, kan een constante beeldverversingssnelheid worden bereikt in real-time, zonder manuele configuratie. De encoder gebruikt de meeste complexiteit om voor iedere *Coding Unit* (CU) in H.265/HEVC de optimale encodeermode te zoeken. De optimale encodeermode is de mode met de laagste *rate-distortion*-kost (RD-kost). De complexiteit van het encodeerproces kan dus worden verminderd door minder encodeermodes te evalueren. De beste encodeermode zal echter wel een extra kost introduceren ten opzichte van de optimale RD-kost. Door een complexiteitsbudget te introduceren zal de encoder voor iedere CU, na-

dat de RD-kost berekend werd van een encodeermode, evalueren of het de moeite loont om te zoeken naar een nog betere optie. Deze evaluatie is afhankelijk van een drempelwaarde die wordt opgegeven door een complexiteitscontroller. De complexiteitscontroller zal trachten om de geschatte afwijking op de RD-kost te beperken en deze evenredig te verspreiden over het beeld. De encoder zal zich automatisch aanpassen aan de karakteristieken in de videobeelden en de gebruikte hardwaremogelijkheden. Hierdoor, is het mogelijk om met complexiteitsbeperking in software-encodering betrouwbare producten te maken. De gepresenteerde methode werd gedemonstreerd met zowel de referentiesoftware (HM) als met een real-time implementatie (x265). Het resultaat toont aan dat de techniek effectief is, snel convergeert naar het juiste complexiteitsbudget en dit met maar beperkt kwaliteitsverlies.

Dit onderzoek werd uitgevoerd in de context van een Baekelandmandaat (vroeger IWT, nu VLAIO), het combineert academisch onderzoek aan de Universiteit Gent en de industriële expertise van Barco. Dit werk heeft geleid tot 4 patentaanvragen, 1 artikel in een internationaal wetenschapelijk tijdschrift en 3 bijdragen op internationale wetenschapelijke conferenties als eerste auteur.

English summary

Video streaming applications are omnipresent in our everyday lives, and hardware-software support for such applications is considered a given in modern consumer electronics such as laptops, tablets, smartphones and televisions. Professional video systems (e.g., security and surveillance, events, education), however, cannot simply integrate consumer electronics due to the higher requirements of professional applications with respect to reliability, latency, and pixel fidelity and quality. The scope of this dissertation is to investigate how consumer electronics and common video compression standards (in particular, H.265/HEVC) can be used to reduce cost and improve interoperability in professional environments, while at the same time still satisfy the requirements customers typically expect from professional video systems.

Consumer electronics typically only provide hardware support for conventional video coding profiles such as H.265/HEVC Main profile, which only allows 4:2:0 chroma subsampled video content. This chroma subsampling produces artifacts that are disturbing in professional video applications featuring screen content (e.g., industrial process monitoring). Therefore, as a first main contribution in this dissertation, we propose techniques to recover as much as possible from chroma subsampling artifacts.

In a first approach, the guided image filter is used to enhance the chroma components of the decoded stream, by using the luma component as the guide. Using the guided image filter with a fixed parameter configuration does not improve the quality of screen content. Improvements of the content are very local and a more adaptive approach is explored. A prediction model is created based on histogram analysis of the luma component. This prediction model is used to configure the radius parameter of the guided image filter. This method works well for screen content with a limited amount of unique colors. The model does not work if more unique colors are used (e.g. gradient or transparent areas), it even introduces new artifacts in this case. Because there is not much similarity between the optimal radius parameter of neighboring pixels in areas with lots of unique colors. This makes it difficult to find a better prediction model. Therefore, it is opted to extend the bitstream with additional information so the decoder can reconfigure the guided image filter. This information will be ignored by third party decoders making the solution standards compliant. With the *Minimum difference offset* (MDO) technique, the encoder only signals to use the guided image filter if there is a large

improvement in the output of the filter. This method prevents the introduction of new artifacts. By configuring the MDO, the user also has the possibility to balance between quality and additional bandwidth. However, this solution needs modifications to both the encoder and the decoder.

The second approach for chroma reconstruction, Guided Chroma Reconstruction, allows that the decoder can also enhance the screen content stream from third party devices (including consumer electronics). This method does not require any additional information from the encoder. This method is, like previous experiments, based on the assumption that for screen content the luma and chroma component have correlation, additionally it also takes the chroma subsample filter into account that was used in the encoder. Firstly, it is assumed that if two pixels have the same luma value, they also share the same chroma value. In a second assumption, it is tested if luma and chroma have a similar structure. Guided chroma reconstruction only adapts a pixel component if there is a high probability that the assumptions are correct. This method provides improvements for both lossless and lossy encoding. The impact on camera captured content is very low as it detects no correlation between the luma and chroma components and therefore does not change the chroma component. As a consequence, there is no need to classify the content before using this reconstruction method.

Since hardware-acceleration is not always available in consumer electronics, as a second main contribution in this dissertation we propose techniques to improve the performance and reliability of video encoding in software through complexity-constrained encoding, in order to meet professional requirements. In current real-time software implementations, the time required for an encoder to encode one frame depends on a range of factors including the characteristics of the video content, the capabilities of the host system, and (manual) configuration settings. This however introduces variability and unpredictability, resulting in occasional frame drops or overprovisioning of system resources. By modifying the software encoder to account for a complexity budget, constant frame rate can be achieved in a real-time software encoder, without manual configuration. Most of the complexity in the encoder is consumed by searching the optimal encoding mode for every Coding Unit (CU). The optimal encoding mode is the mode with the lowest Rate-Distortion (RD) cost. Complexity in the encoding process can be reduced by decreasing the number of encoding modes that are evaluated, but this will introduce a RD cost error.

With complexity constrained encoding, the encoder will for every CU, after calculating the RD-cost of an encoding mode, evaluate if continue searching for a better encoding mode is worth the effort. Therefore it will predict the expected RD cost and spread the RD cost error evenly in the frame. The encoder automatically adapts to the content characteristics as well as the available hardware resources. Therefore, with complexity constrained encoding it is possible to build a more predictable software based video encoder product. The proposed method

is demonstrated in the reference implementation (HM) and a real-time implementation (x265). The results show that the technique is effective, providing fast convergence while incurring only a limited loss in RD performance.

This research has been conducted in the context of a Baekeland PhD grant (formerly IWT, now VLAIO), combining academic research at Ghent University and industrial expertise at Barco. This work has led to 4 patent applications, 1 paper in an international journal, and 3 papers on international conferences as a first author.

1 Introduction

We are all getting used to the comfort of online video streaming services like Netflix [1], Youtube [2], and Vimeo [3]. We can watch a movie anywhere, anytime, and on any device. We can start watching on our computer, seamlessly continue on our smartphone while grabbing a beer in the fridge, and end up in the sofa watching the movie on our smart TV. We can pause the stream for a video call to our family, to congratulate someone on his / her birthday.

To realize such flexibility, today's video streaming services incorporate a wide range of technologies to adapt to the user's environment, to detect and compensate for changing characteristics in the network, varying display resolutions and varying end device capabilities. In addition to the functionalities related to the video streaming itself, the services have managed to become full-blown multimedia platforms that analyze user preferences, and offer recommendations and ratings from other users. To enable such functionality, the service provider stores the user's interactions (e.g., which content has been watched, for what time period), to build a profile of the user's viewing preferences. This provides a fluent, personal overall experience with no or very limited manual configuration needed from the end user.

However, does a similar apparently seamless experience also translate to professional video streaming solutions? Can an operator in an oil refinery view all system information on his tablet while walking around the company? Can a manager attend a meeting from an airport at the other side of the world? Can a surgeon from within a digital operating room call for a second opinion from a remote surgeon who is at home? The answer at the moment is in most cases - no.

This is due to various reasons (including differences in requirements, regula-

Figure 1.1: Barco's new headquarters in Kortrijk, Belgium (the construction of the building was finalized in 2016).

tions, technology adoption speed, etc). In this chapter we will cover these aspects in detail, as it forms the core motivation for the research reported in this dissertation. But since this research has been conducted in collaboration with the company Barco [4], first, we shortly describe Barco's activities in the following section to clarify the industry context. Next, we will explain the requirements and challenges as currently faced in professional video streaming services, and formulate the research questions that lead up to several innovations, described in the following chapters of this dissertation.

1.1 Barco - company introduction

Barco is an international company specialized in connected visualization systems for professional use. Barco is present in more than 90 countries and has currently 3361 employees worldwide. The company headquarters are located in Kortrijk, Belgium (see Figure 1.1).

The company was founded in 1934 as the Belgian American Radio Corporation (BARCO). Initially, as the company name suggests, its activities focused on producing radios that were designed and manufactured with American components in a workplace in Poperinge (Belgium). In 1948, Barco made its first steps in visualization with a multi-standard television (see Figure 1.2). Barco was soon recognized as a leader in televisions, as its televisions supported a wide range of new signal standards. In the late '60s, Barco was among the first companies to introduce color television.

(a) (b) (c)

Figure 1.2: Barco products from the past: (a) one of the first televisions in 1948, (b) a broadcast control from 1967 and (c) healthcare monitors dating back to 1986.

Extending beyond consumer radios and television sets, Barco soon entered the professional broadcast market by producing equipment for control rooms (see Figure 1.2). Based on the success of these products, they expanded their activities to other professional markets. In 1979, Barco made its first steps in projection technologies and since then Barco has been focusing solely on professional markets. In the 80's, Barco was the prime projection technology provider for computer giants such as IBM, Apple and Hewlett-Packard. Back then, Barco pioneered in using core technologies such as Liquid Crystal Displays (LCDs), light-emitting diodes (LEDs), Texas Instruments Digital Light Processing (DLP), and later on, liquid crystal on silicon (LCoS). Barco's portfolio currently consists of a broad range of products and professional visualization solutions, delivering superior image quality in multiple markets. Recently, their focus expanded from cutting edge display and projection hardware incorporating state-of-the-art image processing, to now also include networked visualization and collaboration solutions. More details can be found on the company's website www.barco.com or below.

1.1.1 Company structure and products

Barco is currently organized in three divisions: Entertainment, Enterprise and Healthcare. In this section, more details will be provided on the activities and products of each of these divisions.

The **Entertainment Division** is focused on four markets. The first market contains the products and services related to the cinema business. A key component of any cinema is the projection system. The newest generation of flagship cinema projectors [5] use lasers as a light source. A single projector can produce up to 56000 lumens, which is double the brightness as compared to Xenon lamp-based projectors.

With Barco Escape [6], three screens (each with their own projector) are used

Figure 1.3: A cinema equipped with Barco Escape. Three projectors and screens provide a panoramic viewing experience.

to create a panoramic cinema viewing experience (see Figure 1.3).

For cinema, Barco does not only provide (3D) projection technology but also 3D audio solutions. Auro 11.1 [7] offers a natural and immersive movie experience by placing audio speakers on three different levels. Barco also transforms the theater lobby, by offering a wide variety of screens and software to increase the customer experience by providing additional information (e.g., timetables, availability) or entertainment (e.g., social media interaction).

A second market for the entertainment division, Events & Pro AV, includes equipment to create flexible (i.e., not rectangular) screens. Such screens are typically used during trade shows and large music festivals (e.g., the Eurovision Song Contest). The retail and advertising market is served with display and projection systems that are waterproof and suitable for outdoor use. From the software side, Barco offers the X2O visual communication platform which makes it easy to aggregate different streams of information from different sources (e.g., spread sheets, twitter feeds, videos, etc.) and visualize them in a comprehensive and interactive way.

The last market, Simulation, provides caves or domes with immersive audio and video projection systems for photorealistic large simulators,used in car design, the oil & gas industry and airplane simulation.

Figure 1.4: Barco products for the meeting room.

Figure 1.5: A control room with a video wall (24480x3840 pixels) and multiple operator workstations at Austin Energy, Texas, USA

The **Enterprise Division** provides products for four different markets. In the collaboration market (e.g. the meeting room), Barco offers a wireless presentation system called ClickShare [8]. After inserting the ClickShare (USB) button into their computer (see Figure 1.4), the user can, with a single button-click, mirror his or her laptop screen to the large display or projector in the meeting room. Barco also provides special meeting room projectors, featuring very low noise levels. For control rooms, Barco offers large, seamless video wall display systems, video wall controllers and software to manage all the content. An example of a typical control room is provided in Figure 1.5. In this example, a large video wall with a total resolution of 24480x3840 pixels allows to visualize different video sources, including

output from so-called Supervisory Control And Data Acquisition (SCADA) applications (e.g., for process monitoring), in combination with annotated geographical maps and feeds from security cameras on the site. Based on the current situation, the layout and content displayed on the video wall can be changed flexibly. An operator in a control room can also see the content that is shown on the video wall directly on his personal operator workstation (OpSpace). A range of wall controllers and management software is responsible for configuring all the network streams and converting the streams to the correct format (if required).

Next to these markets, the enterprise division also serves the virtual reality market and the education market. In the education market, Barco focuses on solutions that reduce the gap between students and teachers, for example, through remote participation, improved interactivity and support for new teaching methods (e.g., "flipped classroom").

The **Healthcare Division** provides an integrated approach to patient care across the hospital, improving clinical outcome in many departments via a connected network of display systems. In departments such as radiology and mammography, Barco has a very strong reputation for delivering high end displays. Image quality and consistency is very important for a correct diagnosis, and display performance needs to comply with strict healthcare regulations. To achieve this, the quality of the displays is continuously monitored and dynamically recalibrated in real-time. Malfunctions are automatically reported using a software platform called MediCal QAWeb [9].

In addition to high end monitors, Barco introduced Nexxis [10] as a video-over-IP solution for the integrated Digital Operating Room (DiOR). An example of a DiOR can be seen in Figure 1.6. In a DiOR, content is typically captured using high-end cameras (specifically designed for medical applications). In the case of Nexxis, content is then transmitted in uncompressed format over a 10Gbps IP network, allowing for near zero-latency streaming and visualization and thus perfect hand-eye coordination during surgery. Due to the flexibility of such an IP based solution, the medical staff can assist the surgeon and change the content of the displays on the fly.

Next to the displays for physicians and surgeons, Barco Healthcare also produces interactive patient care systems, such as bedside terminals and other visualization tools to improve a patient's experience in the hospital.

1.2 Trends for professional markets

Video solutions for professional applications are typically slow in adopting new technologies. For example, although digital IP-based systems are the de facto standard in consumer applications, the switch from analogue to digital IP-based

Figure 1.6: A DiOR containing multiple displays showing content from medical equipment and video cameras.

systems is still very recent or even ongoing in some professional applications. For the Digital Operating Room (DiOR), IP based systems were only introduced in Barco's Healthcare market from 2011 onwards.

There are several reasons why new technology is adopted slowly in professional markets:

- A first reason is that professional video systems are very *expensive*. This is due the high requirements towards latency, resolution, and pixel quality, as well as the (custom) hardware and software integration efforts often required. These products are therefore designed to be robust and to long-lasting.

- Another reason why adoption is slow is because many of these systems need to be *always on*. For example, nuclear power plants as well as most chemical factories are always in operation and cannot be temporarily shutdown to perform a software or hardware upgrade. Upgrades have to be planned very carefully to guarantee safety at all times. Some components may only be offline for a minimum amount of time. Therefore, often these systems are extended with new technology or new functionality instead of replacing the older (but still well-functioning) components. To allow upgrades,

compatibility with the older generation of products is mandatory.

- A final reason why it takes a lot of time to introduce new technology is due to *quality and safety standards*. Some products need to be certified first before they can be used. These requirements can be set by the customer or by the government. For example, if a new video compression technology is proposed for the healthcare market, it should first be verified that this compression technology has no influence on the decision making process performed by the surgeon. Providing such proof through analysis and exhaustive tests makes certifying new technologies time consuming and expensive. As a result, the introduction of new technology should be considered very carefully.

We will further elaborate on these reasons and provide additional context for this dissertation through three use case example scenarios, i.e., the Digital Operating Room, Control Rooms, and Corporate Events.

1.2.1 Use case - The Digital Operating Room

Figure 1.6 shows a DiOR. In the operating room the patient is physically operated by a surgeon. As each type of surgery needs a specific setup, all equipment is mobile on itself, or made mobile by attaching it to a flexible mounted to the wall or the ceiling of the room. Video content is captured using medical cameras (endoscopes) or computer generated by medical monitoring devices that register and visualize e.g. heart rate, breathing frequency, oxygen saturation and body temperature. All video sources are connected to a network. Typically by feeding their video output into an encoder dongle that puts the stream on the IP network. In the figure, 6 high quality displays can be seen. These are certified for surgical operations and connected to the same IP network as the video sources via off-the-shelf IP switches. The content presented on these displays can change during the surgery to always show the most relevant information (or it may change for ergonomic reasons). Such changes can be triggered by the surgeon or by an assistant. During keyhole surgery, the physician's only visual feedback is what he sees on his or her monitor. For perfect hand-eye coordination[1] however, end-to-end latency should be minimal as significant deterioration in user experience and clinical performance can be measured from 105 ms latency upwards [11]. For marketing reasons, the maximum required latency have been reduced to 20 ms.

In the past, video sources and displays were hard wired. If a source needed to be displayed on a different display, the only option was to change cables or to use matrix switches. With ethernet-based IP technology, sources and displays

[1] For example, in laparoscopic surgery, the surgeon does not have a direct view on the area of interest but instead (s)he is looking at a display that visualizes the output of medical cameras inside a patient's body.

	HQ	LAN	WAN
Network	Owned	Managed	Unmanaged
Bandwidth	10 Gbps	1 Gbps	1-10 Mbps
Latency	20 ms	80 ms	300 ms
Quality	mathematically lossless	visually lossless	lossy
Hardware	Custom	COTS	COTS
Video Coding Standards	Proprietary	Standardized	Standardized
Cost	$$$	$$	$

Table 1.1: Summary of the requirements for different network domains based on different use cases (DiOR, Control Rooms and Collaboration).

are now connected on a logical layer. Providing a logical connection between the devices simplifies the installation and improves flexibility: the system can now be reconfigured without the need to physically swap cables. Obviously, it is not allowed that the image quality has influence on the decision making process of the surgeon. Therefore, a product that uses compression has to be validated before it can be used in the operating room. This validation is also mandatory for lossless video compression techniques. To avoid the expensive validation process, video sources are not compressed but instead streamed in raw format over high-speed IP connections (e.g., 10 Gbps copper or fiber cables interconnected using off-the-shelf IP switches). This results in mathematically lossless image quality.

The requirements regarding latency and quality are very high within a DiOR. We will therefore label this domain as the High Quality (HQ) domain (see Table 1.1).

In addition to local IP streaming within the DiOR, the logical next step is to *break-out* from this network. For example, an anesthetist within the hospital may be interested to follow multiple surgical operations simultaneously providing feedback to the surgeon or medical staff whenever needed. In a second scenario, the surgeon inside the DiOR might be interested to hear the opinion of a more experienced colleague. This more experienced surgeon does not need to be physically present within the DiOR anymore. He or she can be present anywhere in the hospital, but still assist the surgeon by providing video and audio feedback. The remote expert does not need sub-frame latency since he / she is not performing the actual surgery. Therefore, latency requirements in such scenarios can be somewhat less strict. Typically up to 80ms is considered acceptable here.

Due to cost constraints, it is not always possible to equip every doctor in the hospital with dedicated equipment to follow surgical operations. It would there-

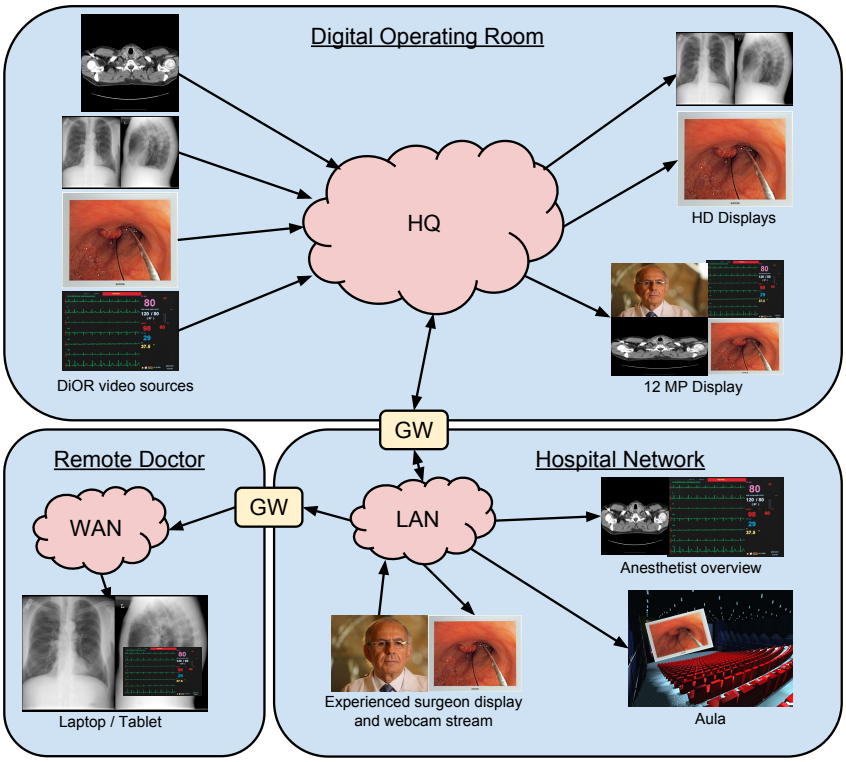


Figure 1.7: Illustration of the different network domains in a digital operating room. "GW" stands for gateway.

fore be interesting if one could use commercial off-the-shelf (COTS) hardware to communicate with the custom infrastructure within the DiOR.

A third scenario includes streaming the surgery live to an aula in the hospital or university campus for educational purposes potentially even with a feedback channel, to allow students to ask questions during live surgery. A diagram of this use case is illustrated in Figure 1.7. The LAN domain includes the managed networks in the hospital and at remote sites. Remote sites are connected to the hospital network with a Multiprotocol Label Swithcing (MPLS) leased line. This leased line ensures a connection between the sites with high reliability and fixed bandwidth. As the content on the LAN network is not used for diagnosis but only for reviewing and providing advice, lossless video compression is not a requirement as such.

These new use cases only became possible with the advent of IP based video communication, allowing DiOR video sources and displays to break out to other networks. Transforming the DiOR video system from dedicated video cabling to an IP-based one in the HQ domain, does not automatically guarantee compatibility

with the rest of the hospital network, in the LAN and WAN domains.

Current hospital networks typically use 1 Gbps or 100 Mbps networks, which does not provide enough bandwidth to transport the uncompressed video streams from within the DiOR. Moreover, the hospital network is also used for other network traffic such as email, internet browsing, telephony, etc. However, this should not be a problem as it is assumed that the IT department of the hospital can configure and scale the hospital network so it can correctly transfer video streams. A GateWay (GW) device should be used to connect the HQ area with the LAN area. In this use case, due to the complexity of transcoding, one gateway device is an encoder that transcodes an uncompressed network stream in a compressed video stream. Similarly, a transcoder gateway device is needed for every compressed video stream that enters the HQ area. These gateways are an expensive part in the whole professional video system. As one gateway device can typically only handle one stream at a time, the number of installed gateways directly imposes limits on the flexibility and number of simultaneous sessions.

In addition to video streaming within the hospital, a logical next step is to enable communication with experts who are outside the hospital managed LAN of the hospital but instead are connected to the internet[2] (e.g., at home, or in a different hospital), using a PC, tablet or smartphone. Since network connectivity is considered unreliable in those case, the service level can only be best effort. In these scenarios too, the doctor is using consumer electronics with limited audio and video communication capabilities. Still, a maximum latency of 300 ms is allowed in to have fluent interactive communication. For wider acceptance and usage, it is impossible to require specific hardware features (e.g. hardware acceleration for video encoding or decoding, minimum CPU specs, etc.). Physicians and hospital IT managers expect easy integration and the best possible performance, given the already rolled out hardware and Bring Your Own Device (BYOD). It is also expected that the applications can be installed from the device specific app store (e.g. iTunes for iOS, Windows Store for Windows 10, etc.) or can even be used without installing any dedicated package, but from within the browser instead (e.g. web applications). This domain (in Table 1.1 it is called the WAN domain) is the most challenging to support since capabilities of the consumer electronics (PCs, laptops, tablets, etc.) vary a lot.

As explained in this use case, all network domains have different capabilities and requirements. Table 1.1 gives an overview of the differences in requirements between the three domains. The first domain, the HQ domain or the operating room has the strictest requirements. The equipment in this domain is owned by the system integrator and is only used for video streaming. The second domain is

[2]Security is considered a key component of Barco's networked professional visualization solutions, in healthcare and other markets. However, it is out-of-scope of this research project and therefore security details will not be described in this dissertation.

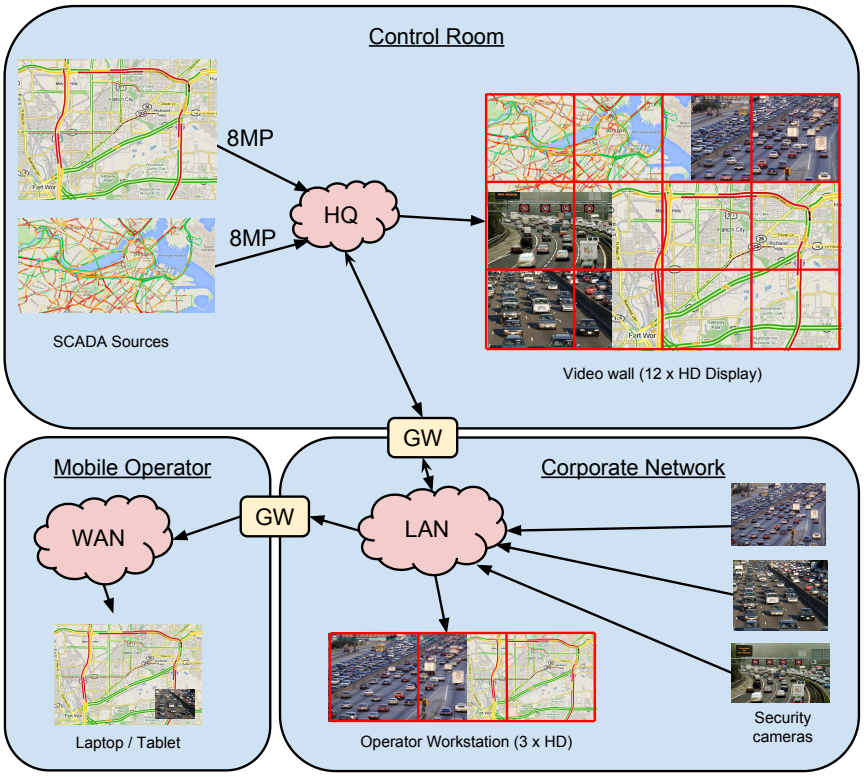


Figure 1.8: A control room with the three network domains.

the LAN domain, this is the company / hospital network that is managed in-house and hence can be configured appropriately between the DiOR and the LAN IT administrators. This network can also be extended with dedicated (e.g. leased) network connections to link sites from the same hospital at different locations. In the LAN domain, video processing equipment is a mixture of dedicated hardware and COTS servers or consumer electronics (BYOD). The last domain is the WAN domain. In this domain there is no control over the network itself, so its service is considered best effort. In this domain, consumer electronics are used.

1.2.2 Use case - Control Rooms

Another use case to illustrate the three network domains is the control room. Control rooms can be installed for different purposes, but their main added value is in providing a clear overview and managing large and complex situations, processes or infrastructure. For example, in a nuclear power plant or a chemical refinery, the large video wall inside the control room will be used to monitor and visualize

temperature levels, fuel gauges, rotation speeds, valve positions, etc. Events like unexpected pressure increase, may trigger layout changes, aiding the operators in making the right decisions. Control rooms are also at the heart of all monitoring and control activities required for smooth airport operation, including airplane air and ground traffic, passenger flows, luggage handling, etc. Also, hundreds of camera feeds from all around the airport can be visualized on the large video wall within the control room for security and surveillance.

A control room is used for process monitoring and control, visualizing all parameters of interest in a nuclear power plant or oil refinery. Events (e.g., unexpected pressure increase) may trigger certain layout changes to aid the operator in making the right decisions. In airports, control rooms are used to monitor and control air traffic, providing a global view on all important activities. Also, several camera feeds from within the airport could be visualized on a large video wall within the control room for security reasons.

Another example of a control room environment can be found in traffic control, as shown in Figure 1.8. In the HQ domain, Supervisory Control and Data Acquisition (SCADA) applications generate graphics about the status of large and complex processes or infrastructure. In this case, the SCADA application will receive data from many different sensors (e.g. traffic lights, traffic counters, variable message signs, etc.) and create a visual high resolution representation of this data. Due to legal and commercial constraints, the raw data itself is typically not accessible, meaning that video walls / controllers, etc. have to operate on the graphics generated by the SCADA applications only.

The display content generated by these SCADA applications is very different from the content generated by security cameras. The SCADA content can have very high resolutions (e.g. 22 x HD and beyond) but still contain details like a blinking alarm box with a line thickness of only 1 pixel wide. These extreme resolution and fidelity requirements make SCADA content very challenging to handle on the network. So in many cases a proprietary solution is used for compression and network streaming. In addition to a more global view available on the video wall, operators typically also have a personal workstation for a more detailed view on specific parts or areas of the process or infrastructure under control. In other words, the SCADA content does not only live in the HQ domain, but should also be available for viewing on multiple other displays. Compression artifacts are to be avoided since they may influence an operator's decision, but some reduction in frame rate is typically considered acceptable. In other words, the operator will always see the correct content (i.e as generated by the SCADA application) but in some cases at a reduced frame rate and / or slightly increased delay.

The company network also feeds into the LAN domain of our visualization system. Think about security cameras or screen scraped desktop PC's and servers. Content from the LAN domain can thus be consumed or displayed on other work-

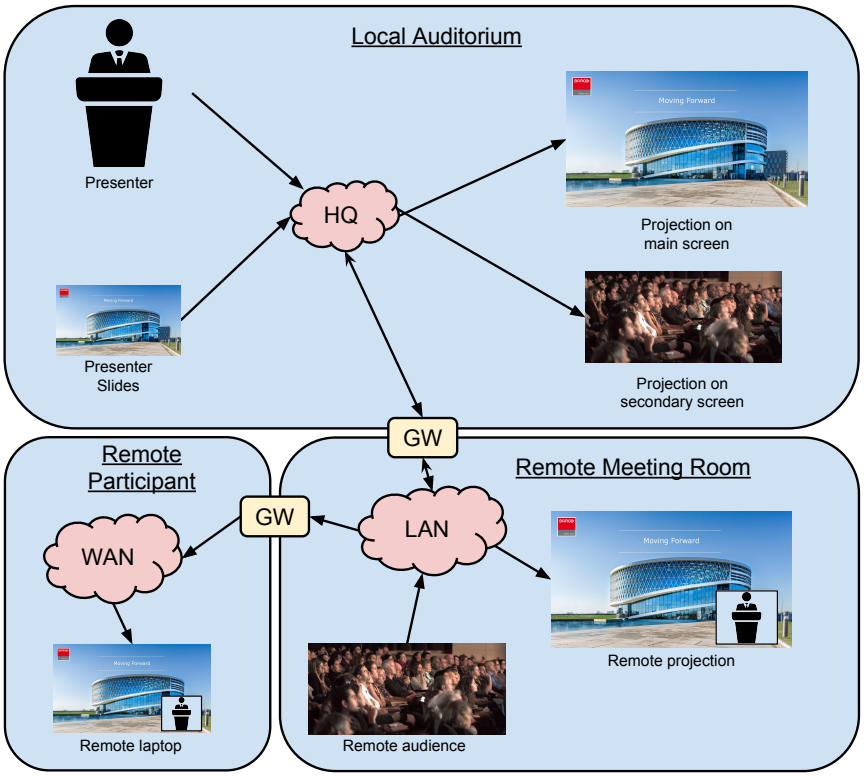


Figure 1.9: A system overview of a company presentation with a local auditorium, a remote meeting room, and a remote participant.

stations in the LAN, but also on the large video wall in the HQ domain. Companies or government instances that monitor road infrastructure, typically span wide areas. When an incident happens, mobile operators go in the field. To assist the decision making process of the remote operators, it's often useful to have access to all content that's available in the control room.

Even when connected via the public internet (i.e., the WAN domain), the operator should have the best quality given current device and network constraints.

1.2.3 Use case - Collaboration

Corporate educational events and presentations (see Figure 1.9) are another typical use case with actors in the 3 network domains. In this use case, a presenter may want to share slides with an audience spread over different company sites. First there is the local audience in the same room as the presenter. The local meeting room or auditorium could be equipped with two projection screens. The first

screen shows the presenter slides. Visual compression artifacts are unacceptable in the local screen mirroring of the slides to the large projected canvas, since the up-scaling would make them very noticeable and distracting for the people in the front of the auditorium. Therefore the local room is considered the HQ area. A second screen in the local auditorium shows the remote audience. The remote audience is physically seated in a remote meeting room on another building or campus of the company. The local auditorium and the remote meeting room are connected through the company network or a leased line. Depending on the required quality for corporate events the network can be temporarily upgraded to a higher bandwidth, but this comes at a financial cost. Therefore the remote meeting room is considered the LAN area. Feedback to the local auditorium is needed to enhance the participation and for asking questions. A remote participant that is located in a hotel, airport or other internet connected location also wants to follow the meeting. Also in that WAN area, the remote participant wants the best quality given the limitations of the internet connection and the capabilities of his device. Typically this is a consumer electronics device such as a laptop, tablet or even a smartphone.

1.3 Research challenges and scope

The previous use case scenarios illustrate that professional video systems can often be logically split into different domains with different requirements (resolved by different technologies). The high requirements towards latency, quality, reliability, etc. of the HQ domain call for dedicated, custom hardware and software solutions while the other domains feature off-the-shelf, standardized technologies with typically less constraining requirements. To enable interoperability between the HQ domain and the other domains, video signals need to be converted by a gateway service which is computationally expensive and further complicates the setup.

A second remark is that designing and producing custom hardware / software solutions for the HQ domain is time consuming. While in the past, developing custom solutions was a necessity due to the lack of consumer-level devices and components, more and more, companies are now exploring the option to integrate off-the-shelf CPU's, GPU's, PCIe plugin accelerator cards, etc. into high-end products for professional applications. This strategy speeds up the development cycle and facilitates re-use and interoperability. It also allows for the deployment of media services, gateways, etc. on cloud infrastructure as a Service (IaaS), where dedicated hardware is not available at all. However, the challenge results in tailoring the hardware / software platform so that it can still address the high demands of the HQ domain.

In the next chapters of this dissertation, we will describe how the above challenges can be addressed by following innovative contributions:

- In Chapter 3, techniques are proposed to add a software layer to off-the-shelf hardware accelerated chroma down- and upsampling (as typically done in GPUs) in order to improve pixel quality for screen content to such extent that it matches the expectations within the HQ domain, where professional solutions are currently typically still deploying dedicated hardware. A first proposed technique uses a guided image filtering technique to improve the chroma components with the luma component as the guide. The filter is extended by making it adaptive for every pixel. Experiments are conducted to predict the optimal filter parameters based on the luma histogram and to extend the H.265/HEVC Main profile base layer with an optional layer containing those filter parameters. A secondly explored technique recovers parts of the chroma component based on the luma component and the subsampling filter that is used at the encoder side. A 1.63 dB PSNR- and 0.051 SSIM-chroma improvement is measured on average for screen content sequences.

- In Chapter 4, the focus is on optimizing video compression to fully exploit the capabilities of off-the-shelf CPU's. Conventional video coding implementations in software can only be configured statically through a number of presets. As a result, the encoding complexity (and encoding time) varies and depends on the underlying capabilities of the computing platform, as well as the characteristics of the video sequence. This results in inefficient use of hardware. As a solution the encoder is adjusted so that it is complexity constrained, in the sense that it automatically adapts internal settings to meet a given complexity target (expressed in msecs). The technique is demonstrated in two encoders. The results demonstrate fast convergence to a given complexity threshold, and a limited loss in rate-distortion performance: on average 2.84% Bjøntegaard delta rate for 40% complexity reduction.

The technical contributions in this dissertation assume quite some background on video compression and streaming, particularly in the context of H.265/HEVC Main profile as it is expected that this video coding standard will be present on most consumer electronics in the near future. The next chapter, Chapter 2, will provide this technical background.

The research performed in the context of this dissertation has led to 4 publications and 4 patents. Explicit references to these publications will be provided in the different chapters themselves, when applicable.

2 Introduction to video streaming

The previous chapter provided general context and elaborated on the differences in requirements between consumer and professional video streaming applications. Due to these differences, integrating consumer electronics in professional video systems is a challenge.

This chapter provides an introduction to video compression and streaming, to build the context for the remaining chapters of this dissertation describing the main technical contributions of this research.

2.1 Video formats

Before describing the process of video processing and streaming, it is important to know how a video stream is represented. Even within a camera, video signals already undergo some processing and transformations before they reach the video encoder. These steps (as well as associated problems) will be described in this section. In chapter 3 it will be investigated how some of these problems can be reduced or even eliminated.

2.1.1 Colorspace

Figure 2.1 provides an overview of the processing steps performed on a video signal passing from the camera to the video encoder. First, the camera captures the intensity of different color primaries ($R_1G_1B_1$) using different sensors. These primaries are specific to the type of sensor and their properties typically also differ

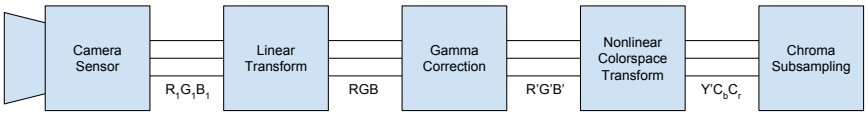


Figure 2.1: Overview of a video signal between the camera and the video encoder with multiple transforms in between.

between brands. As the numerical values produced as output by these sensors do not have a lot of meaning on their own, they need to be converted to interchange primaries in a more general (abstract) reference space, i.e., the RGB domain. In other words, the first processing step within the camera is to perform a tristimulus transform to map the output of the sensors to the RGB domain. In this color space, every color is unambiguously defined. With this information, it should be possible for a monitor to reproduce the exact frequencies captured by the camera. As the perception of light intensity by humans is not linearly related to the amplitude of the color that is measured, the signal is again converted so that it corresponds better to the actual perceived quality. This is called *gamma correction* [12]. As there is redundancy between the different color components of the signal, the signal is transformed to another color space to prepare it for compression. This is typically the Y'CbCr color space. This is indicated as a non-linear transform as the luma component (Y') is a linear transform of the non-linear R'G'B' signal. Therefore, the luma component is not a perfect representation of the actual luminance of the video signal. It is only an approximation. The human vision is more sensible to the intensity of a video signal instead of the color. Therefore a first step, before compression, is to reduce the color information. This process is called chroma subsampling and will be described in more detail in the next section.

2.1.2 Chroma-Subsampling

One of the goals of chroma-subsampling is to reduce the frame size and/or transmission time. The bandwidth can be optimized by reducing the chroma information for a frame. At normal viewing conditions there will be no perceptible quality loss by reducing the color information. To reduce the color information, the video signal needs to be in a color space domain that separates the luminance from the color information. Chroma-subsampling can be done in the Y'CbCr domain but this cannot be done directly in the R'G'B' domain.

There are different possibilities to reduce the chroma information. A first chroma sampling is Y'CbCr 4:4:4 chroma subsampling, in this sampling the chroma components have the same number of samples as the luma component (Y'). This method is used in high-end film cameras, post production or healthcare.

A second common sampling method is Y'CbCr 4:2:2 chroma subsampling.

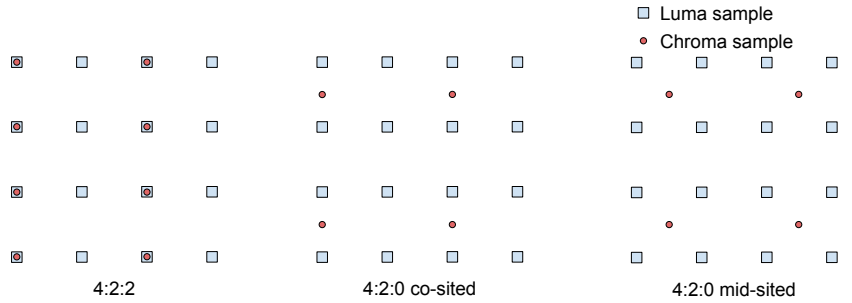


Figure 2.2: Different sample positions between luma and chroma components for Y'CbCr 4:2:2 chroma subsampling and Y'CbCr 4:2:0 chroma subsampling.

Here, the chroma components are horizontally sampled at half the sampling rate of the luma component (see Figure 2.2). This chroma subsampling method reduces the file size of a frame by $1/3$. As this is nearly not visible this sampling method is commonly used in a broadcasting environment, especially in capturing and processing.

The most popular sampling method is Y'CbCr 4:2:0 chroma subsampling. In this method the two chroma components are both horizontally and vertically subsampled at half the sampling rate of the luma component. This reduces the file size of a frame by a factor of two. Y'CbCr 4:2:0 chroma subsampling is very popular in digital video systems because it matches very well with older analog video systems [12] like PAL, NTSC and SECAM. Therefore, this sampling method is supported in nearly all video coding standards and consumer applications like: video streaming, digital television and DVD/Blu-ray. Due to this popularity, it is in most cases the only supported chroma-subsampling on consumer electronics.

Next to the amount of samples in every component, the position of the pixels should also be taken into account. Figure 2.2 shows some common sampling positions for Y'CbCr 4:2:2 chroma subsampling and Y'CbCr 4:2:0 chroma subsampling. In Y'CbCr 4:2:2 chroma subsampling it is most common to position the chroma sample co-sited with the luma sample. In Y'CbCr 4:2:0 chroma subsampling, where there is also a vertical resolution reduction, multiple positions are used in video coding standards. In JPEG [13], MPEG-1 [14], H.261 [15], H.264/AVC and H.265/HEVC the chroma samples are horizontally co-sited with the luma pixel and vertically in the middle of two samples. While in MPEG-2 [16], the chroma samples are both horizontally and vertically sampled between the luma samples (i.e. 4:2:0 mid-sited). It is important when converting signals between different color formats, or when assuming cross component correlation, that the relative positions of the samples for each component are taken into account.

2.2 Video compression standards

To facilitate interoperability (e.g., between devices of different manufacturers), video coding standards are developed and published by standardization bodies. Two of these standardization bodies ITU-T Video Coding Experts Group [17] (VCEG) and ISO/IEC Moving Picture Experts Group [18] (MPEG) have joined forces multiple times to have even more impact. Together they already developed video coding standards as: H.262/MPEG-2 Video [16] and H.264/MPEG-4 Advanced Video Coding [19] (AVC). Video coding standards provided by these organization have a strong impact on the consumer market as they are used in Digital Video Broadcasting [20] (DVB), DVD and Blu-ray and online streaming. Recently, they have created the Joint Collaborative Team on Video Coding (JCT-VC) for development of the High Efficiency Video Coding (H.265/HEVC) standard. Recently, a new organization, Alliance for Open Media [21] (AOM) was created by technology leaders to create a new video coding standard that is royalty free.

Video coding standards define only the bitstream syntax and the decoding process. This means that the decoder knows exactly how to decode the bitstream and how to reconstruct the video. The way the encoder analyzes the video to create the bitstream is undefined. The only requirement is that it should result in a bitstream conforming to the specification. This freedom at the encoder side opens up many possibilities for continuous optimization, and it will also be exploited in chapter 4 to control complexity by dynamically applying shortcuts during the encoding process of a frame. Note that a video coding standard is a specification and not a quality label as such. A naive encoder may create bitstreams that conform to the specification but have poor visual quality (in relation to the size of the bitstream). In contrast, it is also possible to create a bitstream that produces high visual quality but demands a lot of complexity (or memory access) at the decoder. Instead of requiring all tools defined in the video coding standards to be supported in a decoder, some granularity is defined through the use of profiles, levels, and / or tiers.

Profiles define a subset of all possible tools described in the video coding standard. An encoder implementation for a given profile can choose to use or not to use all of the tools in that profile, but it is not mandatory to use all of these tools. However, a decoder must support all of the tools to conform to a given profile. Apart from functional constraints, levels and tiers specify performance-related boundaries. *Levels* are used to indicate a maximum sample rate and bitrate for the bitstream. Some video coding standards (such as H.265/HEVC) define *Tiers* to allow higher bitrates for more demanding applications.

To conclude, to be sure if a decoder can decode a particular bitstream, it is mandatory to know the supported video coding standard, profile, level and optionally tier.

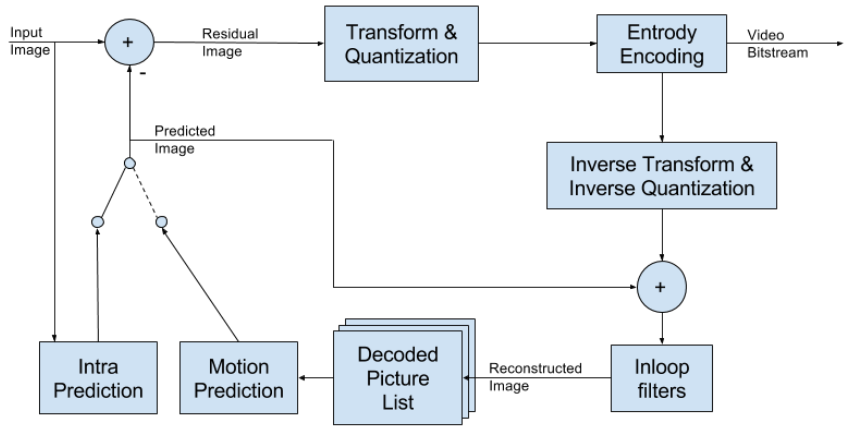


Figure 2.3: High-level overview of an H.265/HEVC encoder.

2.3 H.265/HEVC

The goal of H.265/HEVC, released in 2013, was to provide 50% bit-rate reduction for equal perceptual video quality over H.264/AVC (its predecessor released in 2003). The use cases for H.265/HEVC were the same as for H.264/AVC but the development of the functional elements was driven by two new trends. First, there was a need for higher resolutions, where H.264/AVC is particularly designed for HD content, H.265/HEVC was targeted to support much higher resolutions. A second new trend was implementation specific, as new processors provide more tools for parallel processing, this needed to be taken into account in the design of the standard so that efficient implementations can be developed exploiting the capabilities of modern computer hardware platforms.

The improvements proposed in the next chapters of this dissertation are demonstrated within the context of the H.265/HEVC standard. Therefore, a more in-depth description of the functional components within the H.265/HEVC standard will be provided next. For more details we refer to the specification [22].

H.265/HEVC uses the same block-based hybrid video coding architecture as used in previous video coding standards. A high-level overview of an H.265/HEVC encoder is provided in Figure 2.3. From a high level point of view, the input image is first split into block-shaped regions. For each of these blocks, the encoder generates a prediction, by comparing the block to different blocks at different locations in previously encoded frames. This step is called *motion prediction* or *motion estimation*. The result of this step is (1) the best prediction for the block given a particular metric (such as rate-distortion cost) and (2) a reference / pointer to that prediction (called the Motion Vector, MV). Periodically, or when temporal

correlation is low, the encoder uses previously encoded pixels in the same frame to generate a prediction for the current block. This is referred to as *intra prediction*. When the best prediction is found, the next step is to calculate the residual between the block and its prediction. This residual is transformed from the pixel to the frequency domain, in most cases using the Discrete Cosine Transform (DCT) (and in some cases also additional transforms). Next, a rounding operation occurs, called *quantization*. The goal of this step is to reduce the amount of information contained in the video signal (also called the *entropy*) to enable matching a given bitrate target. Since the human eye is less sensitive to high-frequency detail in an image (e.g., details of textures), quantization typically introduces more loss in the high-frequency components compared to the lower frequencies. The visibility of the artifacts introduced by quantization typically depends on the target bitrate. Artifacts will be more visible on lower bitrates. Finally, quantized residuals in addition to signaling information (regarding motion vectors, coding modes, block structures, etc.) are entropy coded to construct the bitstream that is generated as output by the encoder.

The encoder also performs the reverse quantization and transformation steps to generate the decoded frame. To reduce some of the quantization artifacts, post-processing filters are applied (Inloop filters), such as a deblocking filter and Sample Adaptive Offset (SAO) filter. After this step, the frame is stored in the Decoded Picture List where it can be used in the context of motion estimation for frames that still need to be encoded. Using only decoded frames avoids drift between the encoder and decoder since they both have the same list of decoded frames.

2.3.1 Block structure

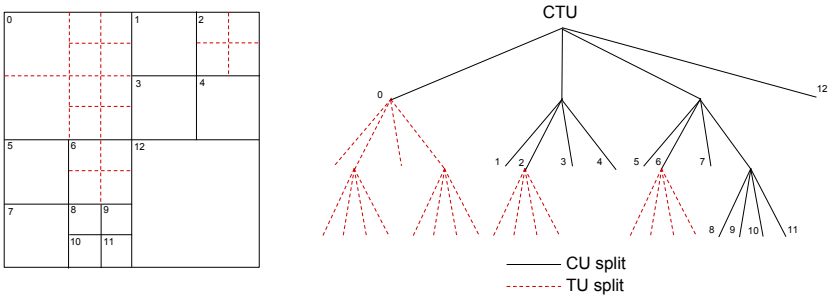


Figure 2.4: Example of a quadtree coding structure, a CTU can be recursively split into CUs and a CU can be recursively split into TUs.

Compared to H.264/AVC, H.265/HEVC has a more advanced and flexible method for block partitioning. Each frame is divided into Coding Tree Units

(CTU) of a predefined size. The encoder can choose the CTU size from 16x16 up-to 64x64 pixels in powers of 2. Each CTU can be recursively split into Coding Units (CUs) using a quad-tree. The minimum size of a CU is 8x8 pixels. Choosing a good CTU structure is very complex due to the huge amount of options. A raster scan order is used to address the CTUs while a Z-scan order is used to address the CUs. This is visualized in Figure 2.4.

A CU represents a block of pixels and contains three Coding Blocks (CB), one for the luma component and 2 for the chroma component. This means that the block structure for luma and chroma components is the same. Each CU can be further split into prediction units (PUs) and a tree of transform units (TUs). Again, a Prediction Block (PB) and Transform Block (TB) represent the color component information of respectively a PU and TU. A CU can be split in 8 different ways into PUs. The shape of the split does not have to be square (which is called an asymmetric split) and splits can be horizontally or vertically. The simplest form is when the PU size is equal to the CU size. In this case, we call it 2Nx2N. The CU can also split in TUs. Here the CU is the root of the transform quad-tree and the smallest leafs of the TU quad-tree can be 4x4 pixels. A TU should, in contrast to a PU, always be symmetric. These TUs are then transformed using a discrete cosine transform (DCT), the intra luma pixels can also be transformed using a discrete sine transform (DST).

In the case of inter prediction, the motion vector is signaled with an advanced vector prediction (AMVP) method. This method uses derivation of several most probable candidates from adjacent and reference pictures. These motion vectors can reference a prediction with quarter-pixel precision. 7-tap or 8-tap filters are used for interpolating this high precision up-scaled version. Like in H.264/AVC multiple reference pictures can be used.

Intra prediction now supports 33 directional modes, planar and DC prediction methods. As with inter prediction signaling, the directional mode is signaled taking into account the mode of the surrounding blocks.

Only Context adaptive binary arithmetic coding (CABAC) is used for entropy coding. CABAC is similar as in H.264/AVC, but has undergone some improvements to improve throughput speed, there is also a reduced set of contexts for lower memory requirements.

The in-loop deblocking filter is integrated within the inter-picture prediction loop but has undergone some changes to make it more friendly for parallel implementations. Also, a Sample Adaptive Offset (SAO) filter can be used in HEVC right after the in-loop deblocking filter. This filter improves the reconstruction of the original signal by using look-up tables as well as some additional parameters.

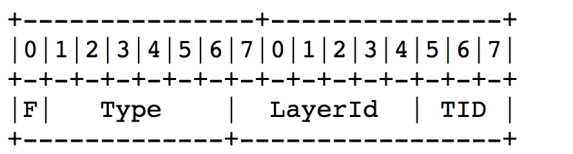


Figure 2.5: Structure of the H.265/HEVC NAL unit header

2.3.2 High-Level Syntax

Similar to H.264/AVC, the bitstream structure in H.265/HEVC makes use of Network Abstraction Layer (NAL) units. The size of the NAL units is typically limited to the Maximum Transmission Unit (MTU) of the actual network. There are two types of NAL units, Video Coding Layer (VCL) NAL units that contain coded pixel information and non-VCL NAL units for metadata. Metadata is mostly used for more than one frame. A special type of a non-VCL NAL unit is a Supplemental Enhancement Information (SEI) message. Such SEI messages contain optional information, i.e., information that is not mandatory to decode the stream correctly. Such information could be color remapping information, or information on the time codes used when the stream was recorded.

The NAL unit header has a fixed size and only contains 2 bytes. Therefore, a Media Aware Network Element (MANE), or media gateway, can parse the header with low overhead and make intelligent decisions on it. Examples could be local repair or redundancy coding, or bitrate reduction by removing temporal layer information if congestion is detected on a network connection. The NAL unit header is shown in Figure 2.5. The forbidden_zero (F) bit must be zero for compatibility reasons. The NAL unit type field (Type) in the header provides the type of NAL unit in the payload. The LayerId field provides the possibility to extend the NAL unit header for scalable extensions. This field must be zero in version 1 of H.265/HEVC. The Temporal Identifier (TID) is used to indicate the temporal layer of the payload of the NAL unit. Every media gateway is allowed to drop all information above a self specified temporal layer, the output stream should still be a stream that is conforming the specification. All NAL units that belong to the same picture should have the same temporal layer. The NALType field in the header provides the type of NAL unit in the payload.

Similar to H.264/AVC, H.265/HEVC uses parameter sets to signal general stream-level or frame-level information. Parameter sets are crucial for decoding and loss of parameter sets often make it impossible to decode the rest of the video data. To enable additional data protection, parameter sets can be sent out-of-band through a different channel (e.g., better protected from data loss).

H.265/HEVC supports three types of parameter sets: Video Parameter Set (VPS), Sequence Parameter Set (SPS) and Picture Parameter Set (PPS). The VPS

is new compared to H.264/AVC and allows to communicate information about the high-level design of a multi-layer codec. The VPS is not mandatory in H.265/HEVC version 1, all information required for decoding is also contained in the SPS. An SPS contains general information that applies to multiple frames in the video sequence. This information is needed to start decoding at an instantaneous decoding refresh (IDR) picture, a Broken Link Access (BLA) picture, or a Clean Random Access (CRA) picture. The PPS contains all information that can change from picture to picture. This information includes initial quantization parameter (QP), a number of flag indicators and tiling information. The highest level header is the slice header, this header contains all information related to the current slice, and has an indicator to which PPS to use. A reference to VPS and SPS is included in the PPS. Each parameter set has support for extensions, this makes it possible that future versions of H.265/HEVC can add extend these parameter sets without breaking backwards compatibility.

H.265/HEVC has four partitioning tools available: regular slices, dependent slices, tiles and wavefront parallel processing (WPP). Regular slices are nearly identical as in H.264/AVC. Regular slices do not allow in-picture prediction outside of the slice. Regular slices do not have entropy coding dependencies over the slide boundaries and can therefore be decoded independently. While regular slices are contained in a single NAL unit, dependent slices can provide segmentation to split a regular slice into multiple NAL units. WPP is a new tool in H.265/HEVC which allows parallelization on a row-by-row basis. Every row of CTUs can be handled in a separate thread. The start of encoding of each row is delayed until the two CTUs (directly above and left above) in the row above are encoded. This allows the encoder to also use intra prediction to code the block. The use of WPP does not increase the number of NAL units, since the output of multiple WPP threads can be stored in the same NAL unit. If WPP is used, the entry point for each WPP layer is indicated in the slice header.

2.3.3 Available implementations

Writing from scratch a fully optimized video codec implementation compliant to the specification is very time consuming. Therefore a popular strategy is to extend existing implementations. This section lists and compares the most important open source H.265/HEVC implementations that are currently available.

HM [23] is the reference software, containing an *encoder* and a *decoder*, that has been used for developing the H.265/HEVC standard specification. Therefore, this implementation incorporates all the features described in the specification. When all features are enabled, the HM encoder delivers the highest compression ratio for a given visual quality. This implementation is not optimized for speed. It is single

threaded and is missing the parallel processing tools (e.g., WPP) as provided by the standard specification. As a result, encoding a single frame can easily take up to one minute and more on a Intel Xeon 2.6Ghz, depending on the settings. The HM implementation is well known, and used as a standard benchmark in research literature to compare different algorithms. In the context of this dissertation, the HM reference software will also be used as a benchmark wherever relevant.

x265 [24] is an optimized *encoder* implementation for H.265/HEVC. It consists of a stand-alone CLI application and a library. MultiCoreWare [25] started this project by integrating the HM into the x264 [26] codebase. This project is open source since July 2013 and is available under both GPL and commercial licenses (similar as the x264 project). It is implemented in C++ and assembly to optimize for speed. It supports a list of Main profiles (e.g. Main, Main10, Main12 and Main Still Profile) and supports bitdepths of 8, 10 and 12 bit. As a big differentiator over HM, it supports encoding using multiple threads. Thread pools can be generated to parallelize motion search, frame based threading and WPP. This makes it possible to build real-time implementations. A demo [27] was given on the 2015 NAB show where MultiCoreWare showed real-time 4K encoding at 60fps on a dual Intel Xeon E5v3 CPU only occupying a single standard rack unit. Although this demonstrates real-time encoding capabilities, the required high end server (with more than 24 cores) is still very expensive. One of the contributions in this dissertation is to study how to adjust this software so that it can run real-time on less expensive systems (as will be explained in Chapter 4).

libde265 [28] is an open source *decoder* implementation of the H.265/HEVC specification. It is released under the GNU Lesser General Public License [29]. It is implemented in C and has SSE optimizations for better performance. The decoder supports WPP and tile-based multi-threading. Therefore, the decoder can run real-time on common desktop PC systems, in contrast to the HM implementation. The decoder can currently only decode Main profile streams. Next to a stand-alone CLI application, plugins are available to support the decoder in most of the popular multimedia frameworks like GStreamer [30], VLC [31], and DirectShow [32]. There is even a Javascript implementation available [33] that provides H.265/HEVC support within any modern web-browser.

FFmpeg [34] is a popular open source library for multimedia handling. Depending on the options, it can be built under the GNU Lesser General Public License 2.1+ [29] or GNU General Public License 2+ [35]. This makes FFmpeg interesting to use in commercial products and/or research projects. The project is mainly developed on Linux, but there is wide support for other operating systems such as Windows, Mac OS X, Android, and iOS. It comes with some applications such

as *ffmpeg*, a general purpose transcoding application. But, it is most famous for its libraries. It comes with a range of libraries that manipulate multimedia files (including audio and video). The most important library for used in the context of this research is *libavcodec*, which contains native implementations of audio/video encoders and decoders (of which most are written from scratch for higher performance). HEVC decoding support was added to FFmpeg in October 2013 via the OpenHEVC decoder project [36]. There is no native H.265/HEVC encoder support in FFmpeg. However, it supports x265 as an external library via its APIs.

GStreamer [30] is a cross-platform multimedia framework. As GStreamer is a framework, it does not contain code to handle video codecs directly but instead wraps different codec implementations. All these implementations can then be used with the same API to create a streaming application. GStreamer is designed around the concept of a pipeline containing different elements that are connected to each other with pads. As such, one of the main advantages of GStreamer is that the elements can be plugged and unplugged flexibly, without the need for modifying the program. This fast development cycle makes it an interesting tool for research in audio and video processing. Next to the architecture for creating pipelines and plugins, GStreamer also provides mechanisms for media type negotiation, synchronization and video conversion. GStreamer is distributed in different packages, the core library (”gstreamer” and ”gst-plugins-base”) and the elements in ”gst-plugins-good” are licensed under the LGPL license (which is interesting also for commercial products). Other elements can have less flexible licenses (like GPL [35]), so care should be taken when incorporating these elements in commercial products.

2.4 Network streaming

In Section 1.2.1 it is explained that, to reduce the system cost, the functionality of a gateway device should change from a encoder/transcoder device to a network stream forwarding device. With the extension of the NAL unit header (see Section 2.3.2) in H.265/HEVC, a gateway device can inspect the bitstream by only parsing the NAL unit headers. However, having a NAL unit header is not enough to build real-time streaming video systems, some essential information like timing information is missing from the bitstream. One of the most used protocols to provide this additional information is the Real-time Transport Protocol (RTP). RTP is briefly introduced in this section to understand how additional information can be filtered from the bitstream in Chapter 3 and latency can be reduced in Chapter 4.

RTP [37] is a network protocol for transferring real-time data over IP networks. The protocol is developed by the Audio/Video Transport working group in Internet Engineering Task Force (IETF). A first version was released in 1996 and

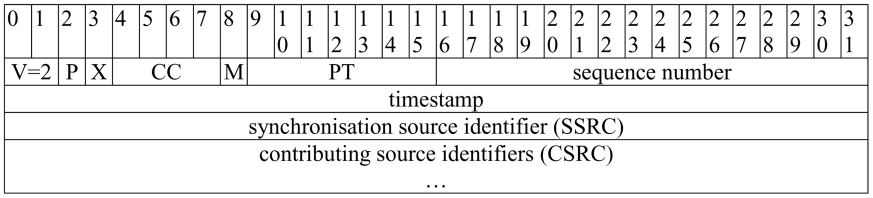


Figure 2.6: Structure of the RTP header

superseded by a second version in 2003. One of the main problems with network streaming is the unreliable behavior of the network. RTP provides the information to detect and compensate for network jitter, out of order arrival of network packets and dropped network packets. As the name suggests it also facilitates in synchronizing the clocks between sender and receiver, RTP can measure and compensate the sender clock drift and can reference the stream to an absolute wall time, this enables synchronization over multiple devices and streams. RTP is the foundation for network friendly multimedia streaming. It is widely used in lots of applications including: streaming media, telephony, video conferencing and digital TV.

The RTP specification defines both the RTP protocol and the RTP Control Protocol (RTCP). RTP only handles the media streams, RTCP handles all optional control information. This information includes transmission statistics, synchronization information and quality of service messages. While RTP only defines packets from the sender to the client, RTCP also defines client messages to the server. In this way the sender can adapt its behavior to optimize the streaming quality. To guarantee good bandwidth utilization for the media data itself, the bandwidth of RTCP is limited to 5% of the RTP bandwidth. The sender or client should delay or drop the RTCP message if the bandwidth limit for RTCP is reached.

The RTP header (see Figure 2.6) contains at least 12 bytes. The first few bits are used for versioning (V) and to indicate the size of the RTP header. Therefore, the padding flag (P) and extension flag (X) can be used to indicate that some respectively padding or a header extension will follow the RTP header before the payload data. The format of header extension is defined by the payload type (PT). The contributing sources (CC) field indicates how many sources contributed to the content of this stream, typically this is provided by a mixer in the network if the audio and/or video are from different sources in the network. The market bit (M) can be used to signal significant events, but exact interpretation is again defined by the payload type. RTP supports the possibility to recover network errors. Therefore the RTP header has a sequence number. This sequence starts at a random value, for security, and increments with every RTP packet. The client can based on this number detect if there is a missing packet or correct the order if the packets arrive out of order. The timestamp in the RTP header also starts at a random value and

increments with the amount of samples in the RTP packet. Thus, the timestamp does not provide absolute time information but only information on the relative offset with the previous RTP packet. The payload type (PT) describes the type of the payload data [38]. It can also indicate that the payload type is *dynamic*, in this case it is assumed that the correct payload type is described in the signaling protocol (e.g. SIP, RTSP).

Due to small variations in the frequency of crystal oscillators used to generate a clock in the server and client, there is always drift on those two clocks (i.e. clock skew). This will result in buffer under runs or over runs in the client which will be noticed as stuttering in the video. As the name suggest, RTP is in the first place a time protocol. The goal of RTP is to transmit the clock information from the sender to the client, which should be able to compensate for the clock skew. This is done by time-stamping the RTP packets and sending them in real-time to the client. The first thing the client does when he receives a RTP packet is also putting a timestamp on it when the RTP was received. If enough packets are collected, the client can based on the data gathered calculate the clock skew, delay and jitter. With the clock skew the client can compensate the playback time of the payload data. As stated before, RTP does not provide synchronization to an external clock. However, RTCP provides functionality to couple the RTP timestamp to a common *NTP*[1] time. This assumes that the RTP sender and receiver have a common *NTP* clock which is not in the scope of RTP.

In the optimal case, RTP expects a steady stream with regular intervals between the RTP packets. There are two reasons why there might be a variation on these intervals. A first reason is that the sender is not sending out RTP packets at regular intervals. This can be the case when the sender is not capturing at a fixed framerate, the processing (e.g. encoding) of the frame is variable in time or bandwidth consumption is not constant in time. This last one is especially true for video content where key frames typically use much more bandwidth compared to inter predicted frames. If the source is not capturing content at a fixed frame rate, the sender is aware of this as the capture time is represented in the timestamp field of the RTP packet. A second reason why there might be variation on the intervals between RTP packets is the network behavior. Packet Delay Variation (PDV) or jitter is caused on networks that typically also handle other data packets. Due to packet bursts of other devices in the network, a switch or router can temporarily queue the RTP packets before forwarding it to the next hop in the network. As a consequence the RTP packets could be delayed or send via another network route to the client. In the worst case, RTP packets can get corrupted or lost on the network. A RTP jitterbuffer is used to compensate for this unpredictable behavior. With the sequence number in the RTP header, RTP packets can be reordered in the correct

[1] While the RTP specification uses the name NTP for a synchronized clock, it does not enforce the use of the Network Time Protocol (NTP)

order. When an RTP packet is missing, the jitterbuffer waits a predefined amount of time and simultaneously queues all other RTP packets. If the RTP packet is not received in time, it can mark that a gap in the stream is detected and continue with the queued RTP packets. The time RTP packets can be queued should be set carefully. If the time is too high, a higher constant latency is introduced in the client. It the time is too low, too many RTP packets can be considered missing which can result in more noticeable artifacts. While the latency can be reconfigured on the fly, this is always visible in the decoder.

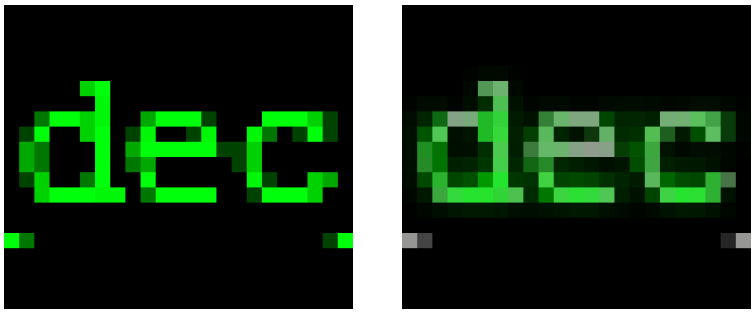
3 Chroma Reconstruction

3.1 Introduction

From a high level point of view, video content can be classified into two main categories, i.e., camera-captured content and screen content.

- *Camera-captured video content* or *natural video content* is generated by cameras and typically contains a relatively wide color spectrum due to the smooth transitions and subtle color variations found in natural scenes. Even when a scene remains apparently static, there are often still small frame-by-frame pixel differences due to minor variations in lighting conditions and other external parameters (e.g., wind) as well as noise generated within the camera sensor itself (e.g., due to thermal or electric variations). Camera-captured video content covers a lot of use cases in broadcasting and consumer applications, and chroma subsampling to 4:2:0 is common practice prior to compression of the video content.

- *Screen content* or *synthetic content* is not generated by a camera but instead it is created by computer systems. This type of content has specific characteristics that often differs significantly from camera-captured content. Often, spatial regions within a video frame feature only one or a very limited set of color values, borders are very sharp (e.g., the border between different windows on a desktop computer), and synthetic elements such as text or lines can only be a single pixel wide. Often, this type of content remains



(a) (b)

Figure 3.1: Illustration of chroma sub-sampling artifacts for screen content. (a) original image (b) artifacts after Y'CbCr 4:2:0 chroma subsampling.

relatively static over time, but major changes can be abrupt. Typical examples including desktop applications (e.g., spread sheets, word processing, internet browsing, presentation slides), visualization of industrial processes, health parameters (e.g., heart rate monitors), and so on.

While 4:2:0 chroma subsampling is common practice for camera-captured content, these artifacts can be too disturbing for screen content [39]. This is particularly true in the context of professional video applications with very high requirements regarding pixel quality and latency (as explained in Chapter 1). As an example, Figure 3.1 illustrates that the text is blurred and colors are not well preserved after chroma subsampling. Therefore, in the context of screen content, compression schemes operating on 4:4:4 format without chroma subsampling are recommended.

Several techniques exist for compressing screen content. During the first months of this research, the H.265/HEVC standardization process was still ongoing, particularly also the various extensions (including screen content coding extensions). Details about these solutions as well as other common strategies for compression 4:4:4 content will be provided in the next section of this chapter.

However, since camera-captured content (and hence 4:2:0) covers a very wide range of consumer-level applications, off-the-shelf hardware acceleration for video coding available in laptops, smartphones, tablets, etc. typically only supports the more common 4:2:0 video coding profiles [40] [41]. Support for Y'CbCr 4:4:4 chroma subsampling video coding is only provided in less efficient and power consuming software. Due to the lower demand, Y'CbCr 4:4:4 chroma subsampling profile enabled (hardware and software) codec implementations are expensive compared to implementations only supporting 4:2:0 chroma subsampling profiles.

Using available hardware acceleration can save power consumption, CPU load, etc., so the challenge is to come up with a technique that reduces or compensates for the 4:2:0 artifacts that are introduced. In this chapter, we propose two approaches. In the first approach (see section 3.4), the encoder sends auxiliary information to the decoder. H.265/HEVC Main profile supports embedding auxiliary information into the coded bitstream by using SEI messages. These SEI messages are optional, meaning that a decoder that does not understand the payload type of these SEI messages can safely ignore these messages. A (modified) decoder can use the information from the SEI messages to configure a chroma up-sample filter for improved quality. This approach assumes customizations to both the encoder and decoder. A second approach (section 3.5) only needs improvements in the decoder device. This enables use cases where the video stream with screen content is encoded on third party hardware and comes from the LAN or WAN network domains. In this second approach the decoder will not use auxiliary packets, only information available in H.265/HEVC Main profile. Instead, cross component correlation will be used to detect and repair chroma subsampling artifacts.

3.2 State-of-the-art

This section provides an overview of existing techniques and standards for compressing screen content. A first subsection focuses on techniques that operate on 4:4:4 content, while a second subsection covers state-of-the-art in techniques that reconstruct 4:4:4 from a 4:2:0 compressed video signal.

3.2.1 Video compression without chroma subsampling

Different video coding standards and techniques are specifically designed for screen content. These techniques are not using chroma subsampling and therefore avoid the typical artifacts described in the introduction of this chapter. However, these techniques cannot be used in the current use case because they are not supported (or hardware accelerated) in consumer electronics or because they lack the flexibility to meet the latency and/or bandwidth requirements in the LAN and WAN domain. Some of these techniques are described in this subsection.

3.2.1.1 H.265/HEVC extensions

After finalization of the H.265/HEVC version 1 specification, JCT-VC continued to work on H.265/HEVC extensions as there was a need to cover more use cases. The requirements of the Format Range Extensions (Rext) [42] covers similar professional markets as described in Chapter 1. The specification also describes some new use cases for Y'CbCr 4:4:4 chroma subsampling. Examples are storage and transmission of professional cameras and wireless display technologies. Another

requirement for the H.265/HEVC extensions was that it should be *similar* to the design of H.265/HEVC version 1. To improve the quality of the chroma components, it has been defined in the Rext specification that every component may be encoded separately. However, it is not adopted in a H.265/HEVC Rext profile as it is considered too computationally complex. To directly encode R'G'B' without colorspace conversion, it is specified that the G' component can be used as the Y' component and the R' and B' component as the chroma components. Next to modifying some existing tools, this extension also introduces some new tools. A first new tool is Cross Component Prediction (CCP), as there might be some statistical dependencies left between the luma and chroma components (especially in the case R'G'B' compression is used), the TB is predicted by a linear model from the luma TB. Another new tool for improved chroma compression is the signaling of the offset for the chroma QP in Adaptive Chroma Quantization Parameter (ACQP). For improved lossless compression, Residual Differential Pulse Code Modulation (RDPCM) allows that the samples, if transform and scaling is skipped, are horizontal and vertical differential pulse code modulated to handle the residual signal. With the Rext specification a BD-rate reduction ranging from 25% to 36% are measured [43] between H.265/HEVC Main 4:4:4 profile and H.264/AVC High 4:4:4 Predictive profile, depending on coding configuration and the content format.

The H.265/HEVC Screen Content Coding (SCC) Extension [44] is also developed in JCT-VC. In the context of this dissertation, a contribution [45] was made to the requirements of this extension. A Call for Proposals (CfP) was issued in January 2014 and the final draft has been added to the specification in June 2015. The main motivator for making this extension was that there where tools proposed in H.265/HEVC version 1 that could perform very well but only for screen content. Screen content can be identified by substantial amount of still or moving rendered graphics, animations or text. Typically there is no sensor noise in screen content. Therefore, this type of content has: larger areas with uniform color, more repeated patterns, highly saturated colors and a limited number of different colors. To take advantage of these characteristics some new tools where introduced. First, Intra Block Copy (IBC) introduced a new CU mode next to intra and inter prediction. The block predictor in IBC can point to block in the already decoded area of the current intra frame to predict the current PU. A second new tool, palette mode, exploits the limited colors by first sending a dictionary with the colors of the current block. A third tool, Adaptive Color Transform (ACT) can remove more inter color component redundancy. ACT can choose to use a different color space on CU level. For example, the current residual can be transformed to Y'CoCg colorspace if required. A last new tool in the SCC extension is adaptive motion vector resolution. Screen content has more discrete motion (i.e. motion is by one or more samples) compared to camera content which has continuous motion. Therefore,

on slice-level there can be switched between full-pel and fractional resolutions for motion vectors. All these tools result in a new H.265/HEVC SCC extension which has more than twice the compression efficiency compared to H.265/HEVC Rext for screen content.

3.2.1.2 Display Stream Compression

Display Stream Compression (DSC) [46] is an example of a video coding standard that supports high bandwidth. The increasing resolution of the displays is causing problems on the physical layer (i.e. the cable) to transport all these pixels. However, Ultra High Definition (UHD) can only be transported over the cable when a low compression ratio (8 bits per pixel) is used. Therefore, the Video Electronics Standards Association (VESA) started in 2012 with the Display Stream Compression task group. Items from the requirements include that the stream should be visually lossless at a constant bitrate of 8 bits per pixel. It should support partial upgrades of the screen with slices. There should also be support for multiple video formats, not only many colorspaces (e.g. R'G'B' and Y'CbCr) but also multiple bitdepths (e.g. 8, 10 and 12 bits). Maybe the most important requirement is that it should be easy and inexpensive to implement in real-time. The first version of DSC is released in July 2014 and is based on Delta Pulse Code Modulation (DPCM) and Indexed Color History (ICH). There is only a single line of pixel storage needed in DSC.

3.2.1.3 Remote desktop protocols

Remote desktop protocols (RDP) are used to mainly do screen scraping and desktop sharing use cases. These protocols provide very high visual quality (up-to lossless) and integrate keyboard/mouse interaction. Examples of remote desktop protocols are Microsoft RDP [47] and the Remote FrameBuffer (RFB) [48] protocol used in Virtual Network Computing (VNC). As the implementation can be pure software, it is inexpensive and it can hook into the operation system. It can for example use the information from the windowing system to detect motion instead of inspecting the captures framebuffers. As the protocols are used over a reliable connection (e.g. TCP), they do not have to care about random access or lost information of the video stream. This also limits the stream to one client per server stream. Typically they contain some low complexity compression tools for screen content like region copy, palette encoding or run length encoding. This makes it very inexpensive to implement in software. It is used in professional markets but these implementations have a lot of problems. First, there is a high latency which provides a bad user experience. This high latency comes from the relative low compression ratio of the simple tools used in combination with a TCP connection. Therefore, RDP is not usable for camera captured content.

3.2.2 Video compression with post-processing chroma reconstruction

Despite various activities on compressing screen content without chroma subsampling, still, hardware acceleration found in today's smartphones, tablets, and PC's typically support only the more common 4:2:0 profiles. Hence, when integrating mainstream CPU's, GPU's, boards, etc. into professional video systems, a system designer has either the option to (1) not use the acceleration (at the cost of additional and potentially heavy CPU overhead) or (2) provide additional logic that allows to recover from the 4:2:0 chroma artifacts that are generated for screen content. The first option will be investigated in the next chapter, the second option in this chapter.

A typical architecture of an encoder/decoder device in a professional market is depicted in Figure 3.2. In most cases the basic computational power is provided by the central CPU (optionally with multiple cores). Previously, the CPU was only used as an orchestrator for all the components on the (embedded) board. Due to the popularity and low prices, the same System on Chip (SoC) devices as used in consumer products are now also used as the main CPU in professional products. Most of the capabilities of the SoC are unused in professional applications. Typically, these SoCs come with hardware acceleration for video encoding and decoding. Depending on the device there could be an additional Image Processing Unit (IPU) (e.g., a GPU, DSP or FPGA) available. In modern SoC devices all the computational power can be combined into a single chip. The IPU is used to combine the output of the decoder with a graphical user interface or other computer generated graphics. The composition is then visualized on a display. Capabilities of this IPU already include image processing operations as chroma up-sampling, format conversions and scaling. As the processing capabilities of such IPU hardware are continuously increasing, some of its computational power is typically not fully utilized in basic playback scenarios. Instead, the CPU often becomes a bottleneck. Therefore, in this chapter we propose to use the freely available computational power in the IPU to increase the image quality for screen content.

If the system needs to support simultaneously Y'CbCr 4:2:0 chroma subsampling and Y'CbCr 4:4:4 chroma subsampling, the sender could simulcast both streams by encoding with two different encoder instances. This solution will provide two independent streams. A receiver could then decode only the video stream that it supports, based on its capabilities. However, this solution is the most complex as it requires two encoder instances and requires more bandwidth.

Another option is to represent the Y'CbCr 4:4:4 chroma subsampling stream as two Y'CbCr 4:2:0 chroma subsampling streams [49], [50] called the *main* and *auxiliary* view. The proposed packing method is designed such that the *main* view is the Y'CbCr 4:2:0 chroma subsampling equivalent of the Y'CbCr 4:2:0 chroma subsampling stream. The *auziliary* view is packed with all chroma information

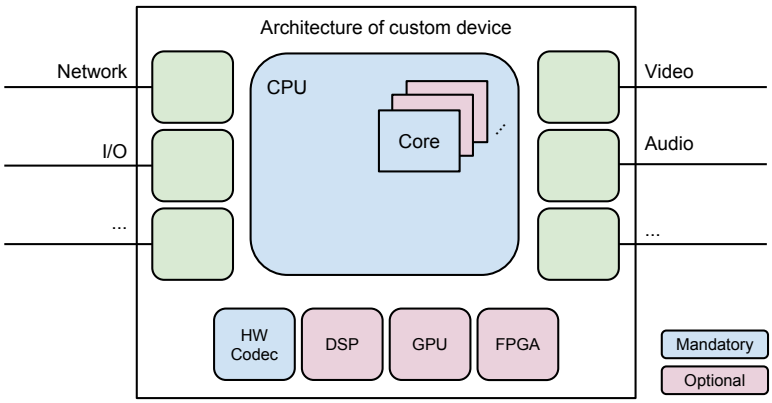


Figure 3.2: Architecture of a typical custom encoder/decoder device. The CPU and hardware accelerated decoder is optionally accompanied by a DSP, GPU or an FPGA.

not presented in the *main* view (i.e. half of the chroma information in the Y'CbCr 4:4:4 chroma subsampling stream). The information is packed across all three components so that both components have similar spatial positioning and motion displacement. If only the Y'CbCr 4:2:0 chroma subsampling stream is required, a decoder can only decode the *main* view. When higher quality is required, the decoder can decode both the *main* view and *auxiliary* view and do the reverse packing method to form a reconstructed Y'CbCr 4:4:4 chroma subsampling stream. Encoding both the *main* view and *auxiliary* view requires two codec instances or one codec instance at double framerate. Typically, hardware encoders do not allow the flexibility to encode two streams or configure a single instance to encode two independent streams. The solution proposed in this section is to create a video stream compliant with conventional Y'CbCr 4:2:0 chroma subsampling profiles (e.g., H.265/HEVC Main profile) and complement it with side information. This side information can then be used at the professional decoder to configure a chroma filter. In this section, the chroma filter is extended by adapting it to the varying spatial characteristics.

Several upscaling filters can be used to upscale the Y'CbCr 4:2:0 chroma subsampling information to Y'CbCr 4:4:4 chroma subsampling. Commonly used filters include the bilinear interpolation filter as used in the chroma upscaling filters for H.264/AVC (2-tap) and HEVC (4-tap) [22]. However, for synthetic content, typically containing very sharp edges and narrow lines, these filters suffer from artifacts (e.g., blurring, aliasing) as they are not designed to handle the characteristics of synthetic content. Adaptive up-sampling techniques that use inter component correlation between luma and chroma as proposed in [51] and [52], reduce these artifacts. These filters combine the information of both the luma and chroma

component to detect edges more accurately. The bilateral filter [53] is an edge-preserving smoothing filter. In general, the output of this filter is the weighted average of the nearby pixels. This will provide high weights to nearby similar pixels. An improvement to the bilateral filter is the guided image filter as introduced by He et al. in [54]. The guided image filter has better behavior near edges, as it can use a guidance image to generate the filtered output. Currently this is one of the fastest edge-preserving filters. Therefore, this filter will be used as basis for upsampling the chroma component.

3.3 Guided Image Filter

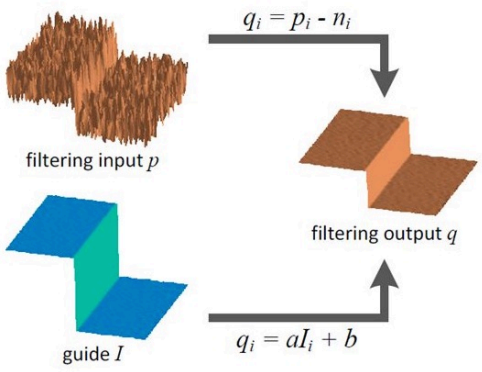


Figure 3.3: The guided image filter uses a guide image to compute the filtering output.

The guided image filter introduced by He et al. in [54] is an edge preserving smoothing filter. The filter transfers structural visual information from the guidance image to the filtered output. It can be implemented efficiently in linear time, regardless of the filters kernel size and intensity range. Applications of this technique are: edge-aware smoothing, detail enhancement and High Dynamic Range (HDR) compression. The idea behind the guided image filter (see also Figure 3.3) is that the output image q is a linear transform of the guidance image I. For each pixel at index i, the output pixel q_i is calculated as a linear transform involving two constants a_k and b_k:

$$q_i = a_k I_i + b_k, \forall i \in \omega_k, \tag{3.1}$$

where a_k and b_k are constant for a window ω_k (containing r by r pixels) centered at pixel k.

The output image q is also defined as the filtering input image p subtracted by noise or textures n:

$$q_i = p_i - n_i, \forall i \in \omega_k. \tag{3.2}$$

The linear coefficients (a_k, b_k) are defined by minimizing the difference between q and p. A regularization parameter ϵ is introduced for penalizing large a_k. Specifically, the given cost function is minimized:

$$E(a_k, b_k) = \sum_{i \in \omega_k} ((a_k I_k + b_k - p_i)^2 + \epsilon a_k^2). \tag{3.3}$$

The solution can be obtained by linear regression (with cov, var denote the covariance and variance respectively):

$$a_k = \frac{cov(I, p)}{var(I) + \epsilon}, \tag{3.4}$$

$$b_k = \bar{p}_k - a_k \bar{I}_k. \tag{3.5}$$

Because pixel q_i can be computed for all overlapping windows ω_k that cover i, the average of all such q_i's is used in the output image q:

$$q_i = \bar{a}_i I_i + \bar{b}_i. \tag{3.6}$$

The filter can be configured with two parameters: the radius parameter and the regularization parameter ϵ. The radius parameter R defines the size of the window ω_k. The radius should be large enough to transfer the texture, but a large radius will also provide more blurry output if no covariance is found between the guide and the input image. The ϵ parameter can enforce that the output is smoothed, even if there is high covariance.

3.4 Adaptive Guided Image Filter

In the research in this dissertation, the guided image filter will be used as the SCC chroma filter. The filter is applied to each chroma component (Cb, Cr) of a decoded image frame, where the luma component acts as the guide. Since screen content typically has sharp edges, where the slope spans only a few pixels, only limited artifacts are generated with down- or up-sampling for smooth transitions. Additional smoothing is not necessary so the regularization parameter is not used (e.g. $\epsilon = 0$). The remaining challenge is therefore finding the optimal radius parameter to configure the guided image filter. Finding the optimal radius R for a

Sequence	Anchor	Radius 3		Radius 5		Radius 7		Radius 9		Radius 11	
	PSNR	PSNR	Gain	PSNR	Gain	PSNR	Gain	PSNR	Gain	PSNR	Gain
sc_map	38.45	38.02	-0.43	37.53	-0.92	37.06	-1.39	36.69	-1.76	36.34	-2.11
sc_web_browsing	34.58	34.21	-0.37	33.95	-0.63	33.93	-0.65	33.90	-0.68	33.88	-0.70
sc_wordEditing	34.62	34.62	0.00	34.18	-0.44	33.73	-0.89	33.36	-1.26	32.98	-1.64
sc_SlideShow	46.17	46.63	0.46	46.68	0.51	46.17	0.00	45.57	-0.60	44.95	-1.22
sc_programming	34.92	34.81	-0.11	34.16	-0.76	33.46	-1.46	32.94	-1.98	32.50	-2.42
sc_ppt_doc_xls	33.91	33.98	0.07	33.68	-0.23	33.36	-0.55	33.04	-0.87	32.75	-1.16

Table 3.1: The average PSNR-Chroma gain for guided image filtering shows that there is no objective quality improvement when using a fixed radius.

(a) (b) (c)

Figure 3.4: (a) original fragment with Y'CbCr values, (b) guided image filter input with chroma artifacts, (c) guided image filter output. The guided image filter smooths undesirably around an edge when no uniform positive or negative covariance in the entire window is found.

frame is equivalent to finding the radius that minimizes the distortion between the original image I and the output image q of the filter. This results in

$$R = \arg\min_{r} \sum_{i} (I_i - q_{i,r})^2 \qquad r = 2n + 1, n \in \mathbb{N}, \tag{3.7}$$

where $q_{i,r}$ denotes the output pixel at index i computed with radius r. As a distortion metric, the sum of squared differences (SSD) is used.

To inform the decoder of the selected radius, an SEI message is included in the stream. As the SEI message only contains the single value of the radius and does not change over multiple frames, the increase in bandwidth is negligible.

Table 3.1 provides an overview of the results when a guided image filter with fixed radius for all pixels is used. The Y'CbCr 4:2:0 chroma subsampling output of the decoder is first upsampled to Y'CbCr 4:4:4 chroma subsampling with the 4-tap interpolation filter proposed in H.265/HEVC (using filter coefficients $\{-4, 36, 36, -4\}$). This is the anchor. The other columns in Table 3.1 are the result of filtering the anchor with a fixed radius. The PSNR-Chroma is the PSNR between the reconstruction and the original image averaged for both chroma components. Only in a few sequences there is a small average PSNR-Chroma gain when a radius of 3 or 5 pixels is used, in all other cases there is quality loss. Nevertheless, there are visual improvements but they are not reflected in the PSNR of the full frame. This is because the improvements are local near edges and, in terms of PSNR-Chroma, are compensated by the wrongly smoothed areas as illustrated in Figure 3.4.

The results from using a global fixed radius demonstrate that a more content-adaptive filtering is required to exploit the local benefits of the guided image filter. Therefore, it will be investigated what the maximal improvement can be with guided image filtering. In this experiment, the guided image filter is applied multiple times for each pixel (and each chroma component) with multiple values for

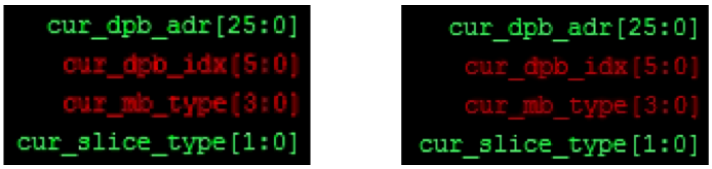
the radius parameter. For every pixel, the radius with the lowest SSD in the output is selected as the optimal radius. The anchor method is not using the guided image filter. Table 3.2 shows the PSNR-Chroma gain when the optimal radius is used. For all sequences there is an improvement in PSNR-Chroma with an average improvement of 2.90 dB. The maximum PSNR gain that can be achieved on these sequences is +3.46 dB for $Video\_Maps\_1$. Figure 3.5 shows a fragment of a frame from the sc_waveform sequence, after filtering with guided image filtering the text is much more readable compared to conventional chroma upsampling. Another example of a local improvement with guided image filtering is shown in Figure 3.6. This figure shows a fragment from the sc_wordEditing sequence, icons in the taskbar look better with guided image filtering. Not only the content type (e.g. a mixture between natural and screen content) but also the colors used in the frame define how much improvements can be made. When lots of black text on white background is used (e.g. like typical document-editing) there is no chroma available so nothing can be improved. However, there are two major bottlenecks with this experiment. A first problem is the computational complexity that is needed for calculating the optimal radius parameter. The second problem is that embedding the optimal radius parameter for each pixel into the bitstream can have high impact on the bitrate of the video stream. Two approaches will be explored to solve these problems: histogram analysis and Minimum Difference Offset (MDO). MDO will be described in next section, first the histogram analysis method will explained.

	Anchor	Optimal radius	
Sequence	PSNR	PSNR	Gain
Office_Scada_1	37.62	41.02	3.40
sc_cad_waveform	24.02	27.11	3.09
Video_Maps_1	38.45	41.91	3.46
sc_programming	34.92	38.20	3.28
sc_video_conferencing_doc_sharing	31.91	33.66	1.75
sc_wordEditing	34.62	37.03	2.41
Average			2.90

Table 3.2: PSNR gain when, for each pixel and chroma component, the optimal radius parameter is selected to configure the guided chroma filter.

3.4.1 Histogram-based analysis

One approach to avoid additional bandwidth for the radius parameter is that the decoder can predict the optimal radius parameter value. Therefore, it will be investigated if the optimal radius value can be predicted from analysis of the luma



(a) (b)

Figure 3.5: Detail from first frame of sc_cad_waveform sequence showing the anchor method (a) and the adaptive guided image filtering method (b).

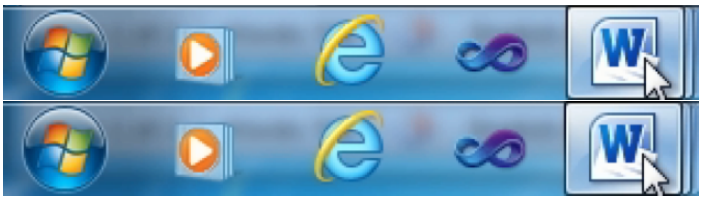


Figure 3.6: Detail from sc_wordEditing sequence, top image (PSNR=37.95 dB) is unfiltered while bottom image (PSNR=37.54 dB) is filtered with adaptive guided image filtering.

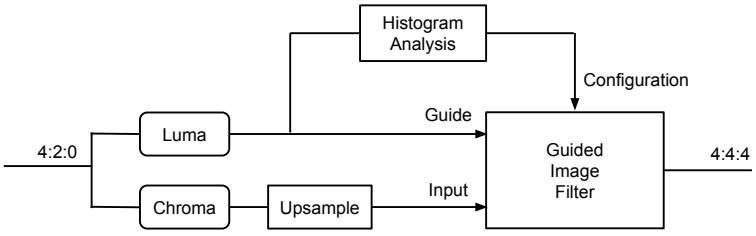


Figure 3.7: A histogram analysis of the luma component is used to re-configure the guided image filter for each pixel.

component. An overview of the experiment is shown in Figure 3.7. The guided image filter is extended with a histogram-based analysis tool to define the radius. A local histogram is created in the learning phase for each pixel in the luma component. This histogram, a square window around the pixel with a fixed radius (11 pixels), is analyzed by taking three parameters into account (i.e., the number of unique luma values, how many times these luma values are used and the difference in luminance between 2 luma values). The filter is applied for all radius values (from a radius value of 1 up to the fixed radius of the histogram). Once all luma pixels are analyzed, the results are collected in a lookup table. This table allows to select the best radius for a set of pixel characteristics, and on average to generate the best output of the filter (PSNR based). In the following tests the same content is used in the training phase as in the evaluation phase, creating the best table for this limited set of sequences.

3.4.1.1 Results

	Anchor	Histogram analysis	
Sequence	PSNR	PSNR	Gain
Office_Scada_1	37.62	38.91	1.29
sc_cad_waveform	24.02	25.88	1.86
Video_Maps_1	38.45	39.39	0.94
sc_programming	34.92	35.45	0.53
sc_video_conferencing_doc_sharing	31.91	32.18	0.27
sc_wordEditing	34.62	34.59	-0.03
Average			0.81

Table 3.3: PSNR Gain after dynamically changing the radius parameter of the guided image filter with histogram based analysis of the luma component.

Table 3.3 shows the PSNR-Chroma Gain after dynamically changing the radius parameter of the guided image filter with histogram based analysis of the luma component. The results show that this method is working well for content where large areas only have a small amount of colors. Examples are Office_Scada_1, sc_cad_waveform and Video_Maps_1 sequences with respectively 1.28 dB, 1.86 dB and 0.94 dB PSNR gain. However, more modern computer generated content has more unique colors in the frame (e.g. like the taskbar in windows having transparent and gradient colors). Therefore, this method does not work well on these type of sequences. On a sequence like sc_wordEditing this method does not work al all (i.e. a PSNR loss of 0.03 dB).

The results show that histogram analysis of the luma component only works well for specific screen content. It does not work on all screen content types.

Because, if a lot of transparency or anti-aliasing is used, neighboring pixels can have completely different optimal radius parameters. Therefore a second method focuses not on predicting the radius parameter but it focuses on reducing the required bandwidth that is needed to transmit the radius parameters for each pixel.

3.4.2 Minimum Difference Offset

In this section, it is proposed to embed the radius parameter of the guided image filter into the bitstream. Figure 3.8 shows an overview of the proposed system with a third party decoder and an SCC optimized decoder. The Y'CbCr 4:4:4 chroma subsampling input stream is first downsampled before encoding. The encoder generates the Main profile stream and the side information to assist the adaptive SCC chroma filter. This information is carried in the original bitstream. A third party decoder ignores this enhancement information and decodes the stream as Y'CbCr 4:2:0 chroma subsampling. The SCC optimized decoder first decodes the stream with an internal Y'CbCr 4:2:0 chroma subsampling decoder. Note that this decoder can be hardware accelerated. Once the decoded stream is upsampled, it is provided to the SCC chroma filter. To optimize the output, this filter uses both the reconstructed stream and the additionally transmitted side information.

The decoder needs information on which radius parameter it should use for each pixel. To solve this problem, the optimal radius parameter at every pixel should be found at the encoder side, which is computationally complex. Furthermore, transmitting the optimal radius for every pixel position to the decoder requires additional bandwidth. Therefore, three steps will be defined to form the proposed solution and provide a good balance between bandwidth, quality and encoder/decoder complexity:

1. Minimum difference offset (MDO): In many cases, radius r results only in an insignificant (and unnoticeable) difference between the pixel J_i at the input of the filter and pixel $O_{i,r}$ at the output. However, calculating the pixel value to compensate for this insignificant difference requires a high computational complexity at the decoder. A threshold (labeled MDO) is defined such that a radius is only selected when the quality improvement is equal to or greater than this threshold. This allows to reduce complexity with a controlled impact on the visual quality, i.e.:

$$D_i(r) = |I_i - O_{i,r}| \tag{3.8}$$

$$\Delta_i(r) = |O_{i,r} - J_i| \tag{3.9}$$

$$R_i = \begin{cases} \arg\min\limits_r D_i(r), & \text{if } \Delta_i(r) \geq MDO \\ 1, & \text{otherwise} \end{cases} \tag{3.10}$$

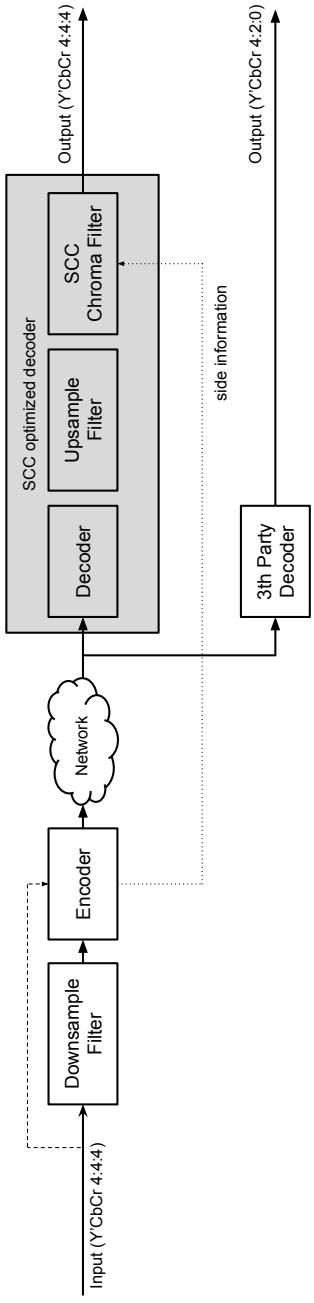


Figure 3.8: Proposed system overview with a SCC optimized decoder and a third party decoder (anchor).

2. Radius option reduction: Similarly, different radiuses can have unnoticeable differences. If only radiuses with significant impact on quality are selectable, complexity at the decoder decreases. In many cases only 2 radiuses are significant, including $r = 1$ where the filter is disabled.

3. JBIG2 encoding: JBIG2 [55] is a compression standard for lossy and lossless bi-level images. Once the number of possible radius values is limited to two, a mask is created for the optimal radius. As both text and more generic content are present in the mask, JBIG2 can use its model-based coding and neighbor based coding tools, respectively. Additionally, JBIG2 has also support for finding similarities between multiple pages. This feature can be compared with inter coding in recent video compression standards. In the proposed system, multi pages are not used, since the video stream is encoded using only intra frames.

3.4.2.1 Results

The proposed techniques are evaluated based on the common test conditions from the development of the range extensions within JCT-VC [56]. The down- and upsample filters used in our system are those used in the development of the scalable extensions of HEVC. The same test sequences from the test conditions are used. As anchor the HM-12.1-RExt-5.1 branch is used, which generates the downsampled Y'CbCr 4:2:0 chroma subsampling streams as well as the original Y'CbCr 4:4:4 chroma subsampling streams. Since industrial applications demanding high quality are targeted, the Y'CbCr 4:2:0 chroma subsampling base layer stream should have limited distortion. Therefore, lossless and super high tier (all intra) configurations are used in our experiments.

RD curves of the streams with enhancement information are presented in Figure 3.9. The streams are compared with the anchor. The radius parameter values are limited to $r \in \{1, 9\}$ and the enhancement information is compressed with JBIG2 in an SEI message. The maximum quality improvement that can be achieved with $MDO = 0$ is a 3.48 dB gain for 36.03% extra bandwidth ($QP = 12$). If the MDO is increased ($MDO = 8$) the quality gain drops to 1.00 dB for only 4.03% extra bandwidth. Adapting the MDO can fine tune the RD-curve for quality or bandwidth.

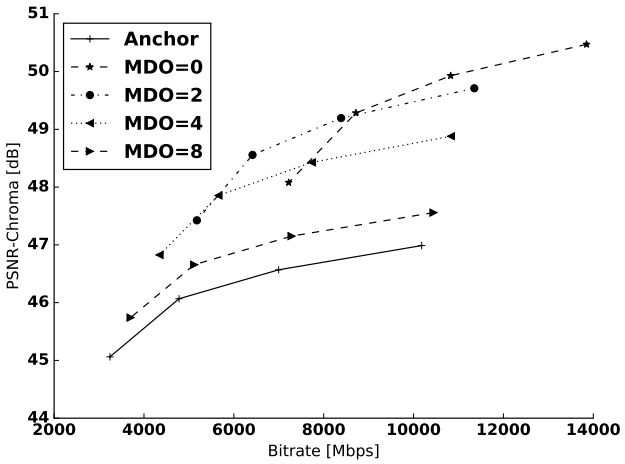


Figure 3.9: RD-curves for SlideShow sequence with QP values of 12, 17, 22 and 27

3.4.3 Conclusion

In industrial applications, the bandwidth of video compression is of a lesser importance than the required quality. Within these constraints, a guided image filtering based technique is proposed to improve the visual quality for screen content applications. This is done by adaptively filtering the chroma information with the optimal radius for the guided image filtering. However, transmitting this information yields an increased bit rate. To limit the decoder complexity, a radius is only used when a minimal difference offset is achieved between the filtered and original pixel. Further reducing the radius space to $r = 1$ (i.e. disable filtering) and $r = 9$ allows to create a mask which can be compressed using JBIG2. This leads to an optimal bit rate for the side information.

3.5 Guided Chroma Reconstruction

3.5.1 Introduction

In the last section about adaptive guided image filtering, it was assumed that the encoder would do some processing on the input stream and embed additional information in the bitstream that helps the decoder to enhance the chroma up sampling process. The downside of this method is that both the encoder and decoder should be modified. In the use cases explained in Chapter 1, this is only possible in the HQ domain. The previous proposal does not work if third party encoder devices are used to encode screen content. Therefore, in this section it will be investigated how a decoder can enhance the quality of the chroma components without additional information generated by the encoder.

Existing adaptive up-sampling techniques that use inter component correlation between luma and chroma are proposed in [51] and [52]. These techniques are designed for natural content and do not take the specific characteristics of screen content into account (e.g. sharp discontinuous edges, homogeneous colors in nearby pixels). Hence, their results show that smoothing is still present around the edges of computer-generated content.

To improve the Y'CbCr 4:4:4 chroma subsampling quality, mainly around the edges, a post-processing step is proposed in this section. Two assumptions are evaluated to improve the chroma component. First, it is assumed that two pixels having the same luma value, likely have the same chroma value. This information can be used to reverse the chroma sub-sampling filter that is used in the encoder and recover the original chroma pixels. The second assumption transfers the luma structure to the chroma component if they have high probability that they share the same gradient structure. When the assumptions could not be validated, the output of a normal up-sample filter is used. No additional smoothing is added to preserve sharp edges. Results are both provided for screen content and non screen content sequences.

3.5.2 Chroma Reconstruction

Figure 3.10 shows an overview of the proposed system. The proposed method for reconstructing the chroma component is designed to be used after the decoder without modifications to the encoder. A luma-based guided technique is used to reconstruct the chroma component. If the luma and chroma component are not correlated, a default upsampling filter is used.

The original stream is captured, for example, through screen capturing and converted to a Y'CbCr 4:4:4 chroma subsampling video stream if necessary. To work with conventional encoders, this stream is then sub-sampled by the encoder device. The encoder will encode streams with a Y'CbCr 4:2:0 chroma subsam-

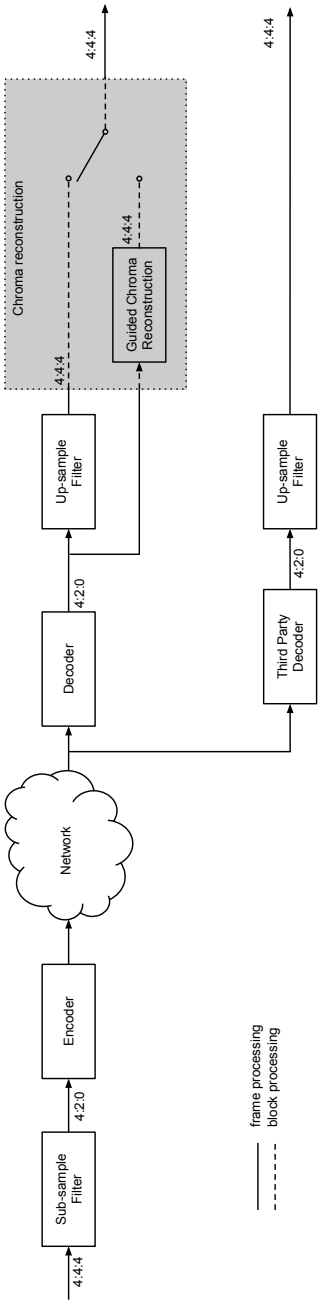


Figure 3.10: Overview of the proposed guided chroma reconstruction method for screen content coding with unmodified encoder. The encoded stream is fully compatible with a third party decoder device (h).

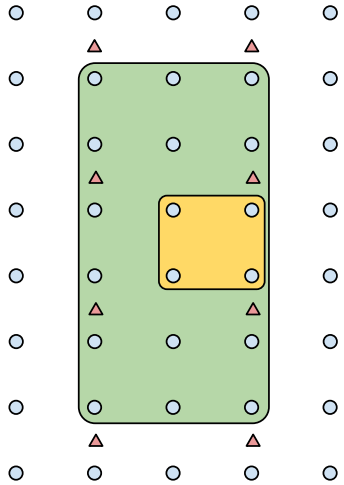


*Figure 3.11: Sliding window **W** (green area) covering the position of 6x3 luma and chroma pixels (circles) at full resolution. Through sub-sampling, 6x3 chroma values are reduced to 2x2 chroma values (triangles). The reconstruction filter generates 2x2 chroma output values (yellow area).*

pling profile, like the HEVC/H.265 Main profile. As said before, it is not necessary to include any additional information in the encoded video stream. After transmission over the network, this video stream is decoded by a standard compliant (hardware accelerated) decoder. This will generate the video stream that will be used as the input for the chroma reconstruction.

The chroma scaler defined in the HEVC/H.265 standard [22] is a 4-taps convolution filter. This standard also defines a nominal position of the sub-sampled chroma pixels. The chroma pixels are co-sited horizontally with the luma pixels and sited between the luma pixels vertically (see Figure 3.11). The chroma reconstruction algorithm in this chapter is based on this filter and chroma pixel position. The algorithm can be adapted to other sub-sample filters and/or chroma pixel positions.

For the chroma reconstruction, every frame is analyzed with a sliding window W. The minimum sliding window (see Figure 3.11) should at least have 2 by 2 pixels in the sub-sampled chroma component. This makes it possible to test and validate the assumptions introduced in the next sub-section. As the four-taps filter uses $r_{0,0}, r_{1,0}, r_{2,0}$ and $r_{3,0}$ to generate the sub-sampled chroma value $c_{0,0}$, the window size of the full resolution components should be 6 by 3 pixels. This is illustrated in Figure 3.11, along with the locations of the chroma and luma values.

Denote the 6 by 3 input luma values as:

$$\boldsymbol{L} = \begin{bmatrix} l_{0,0} & l_{0,1} & l_{0,2} \\ l_{1,0} & l_{1,1} & l_{1,2} \\ l_{2,0} & l_{2,1} & l_{2,2} \\ l_{3,0} & l_{3,1} & l_{3,2} \\ l_{4,0} & l_{4,1} & l_{4,2} \\ l_{5,0} & l_{5,1} & l_{5,2} \end{bmatrix},$$

the 6 by 3 chroma component at full resolution as:

$$\boldsymbol{R} = \begin{bmatrix} r_{0,0} & r_{0,1} & r_{0,2} \\ r_{1,0} & r_{1,1} & r_{1,2} \\ r_{2,0} & r_{2,1} & r_{2,2} \\ r_{3,0} & r_{3,1} & r_{3,2} \\ r_{4,0} & r_{4,1} & r_{4,2} \\ r_{5,0} & r_{5,1} & r_{5,2} \end{bmatrix},$$

the 2 by 2 sub-sampled chroma component as:

$$\boldsymbol{C} = \begin{bmatrix} c_{0,0} & c_{0,1} \\ c_{1,0} & c_{1,1} \end{bmatrix}.$$

Likewise, R' denotes the reconstructed 6 by 3 chroma matrix. Note that in the subscripts of the matrix elements do not correspond directly to the spatial positions of the values (refer to Figure 3.11 for their correspondence).

The goal is to exploit correlation between luma and chroma components. Obviously, when there is no texture in the luma component, there is no texture to be transferred to the chroma component. So these cases cannot be handled by a guided filter. Two correlation assumptions are explored to reconstruct the chroma pixels. These assumptions are explained in detail in the next subsections. To prove that they are valid, the probability of their occurrence is calculated first offline (see Table 3.4). The probability for texture in the luma component ($P(T)$) is measured separately to give a better idea where the cross-correlation can be present.

If an assumption is tested on a window there are three possible outcomes: the conditions for the assumption (CA) are not present, a result is provided in a given range or an invalid result is provided. When both assumptions provide a result with an overlapping range, the intersection of these results will be used. If the two results are invalid or do not have overlapping ranges the output of a common up-sample filter will be used. When the conditions for a given assumption are not present or there is an invalid result, only the output of the other assumption will be used for this window.

3.5.2.1 Assumption 1

The first assumption A_1 is that if two pixels in window W have the same luma value, they also have the same chroma value:

$$l_{n,m} = l_{i,j} \Rightarrow r_{n,m} = r_{i,j} \qquad (n,m) \in W, (i,j) \in W. \tag{3.11}$$

Only when additional conditions (CA_1) are present, the first assumption (A_1) should apply. These conditions (CA_1) require that the current window should at most have four unique luma values (see later in this subsection). This assumption works well for screen content when there are a few unique colors in the sliding window.

Table 3.4 provides the probability $P(T)$ that a window has luma texture and the probability $P(CA_1|T)$ that the conditions for the first assumption are present. Under these conditions, the tested screen content sequences indicate that with a probability $P(A_1|CA_1, T)$ of 98.15% the pixels with equal luma values also share equal chroma values (A_1). This represents on average 35.62% of the tested windows (see Table 3.5). However, it should also be noted that $P(A_1|CA_1, T)$ is very low for natural content. The effect that this assumption is not working for natural content is partly reduced by a low $P(CA_1|T)$. This has a small but negligible negative impact for natural content, as will be discussed in the results section.

The encoder has been using the following formulas to sub-sample the chroma pixels:

$$c_{0,0} = \frac{-r_{0,0} + 9r_{1,0} + 9r_{2,0} - r_{3,0}}{16} + \delta_{0,0} \tag{3.12}$$

$$c_{0,1} = \frac{-r_{0,2} + 9r_{1,2} + 9r_{2,2} - r_{3,2}}{16} + \delta_{0,1} \tag{3.13}$$

$$c_{1,0} = \frac{-r_{2,0} + 9r_{3,0} + 9r_{4,0} - r_{5,0}}{16} + \delta_{1,0} \tag{3.14}$$

$$c_{1,1} = \frac{-r_{2,2} + 9r_{3,2} + 9r_{4,2} - r_{5,2}}{16} + \delta_{1,1}. \tag{3.15}$$

$\delta_{i,j}$ compensate for the rounding errors when converting $c_{i,j}$ to unsigned integer 8-bit values and are in the range $[-0.5, 0.5[$. The decoder has four sub-sampled chroma values $c_{i,j}$ and is trying to resolve twelve original chroma values $r_{i,j}$ using the equations (3.12) to (3.15). Chroma values for pixels with equal luma values are substituted (see equation 3.11). After simplification there should be at most 4 chroma values left to resolve the four equations. Note that the pixels $r_{2,1}$ and $r_{3,1}$ are not taken into account for calculating the sub-sampled chroma pixel values. These pixels should have equal luma values as a pixel from column 0 or 2. The complexity of solving n linear equations is $O(n^3)$. In this assumption there are

Sequence	$P(T)$	$P(CA_1\|T)$	$P(A_1\|CA_1,T)$	$P(CA_2\|T)$	$P(A_2\|CA_2,T)$	$P(CA_1,CA_2\|T)$	$P(A_1,A_2\|CA_1,CA_2,T)$
cad-waveform	50.43	83.52	99.58	35.66	92.96	32.97	95.96
pcb-layout	40.19	86.38	99.99	26.30	92.83	25.33	95.58
ppt-doc	41.33	57.45	97.57	34.19	85.91	24.52	89.35
twist-tunnel	53.28	71.13	99.53	53.94	99.36	36.15	98.76
video-conferencing	62.33	67.97	94.09	37.00	87.61	31.45	87.34
Average	49.51	73.29	98.15	37.42	91.73	30.09	93.40
BirdsInCage	99.96	11.96	0.30	2.92	2.78	0.61	2.45
CrowdRun	100.00	0.10	0.00	0.24	0.17	0.00	0.00
Kimono	99.98	2.25	0.20	3.36	0.34	0.06	0.34
ParkScene	100.00	0.49	0.00	0.29	0.16	0.00	0.00
Traffic	99.91	7.81	1.31	3.67	3.82	0.18	17.58
Average	99.97	4.52	0.36	2.09	1.45	0.17	4.07

Table 3.4: Probability for texture T, conditions for assumptions CA_i and assumption A_i in a given window

Sequence	P_1	P_2	$P_{1,2}$
cad-waveform	41.94	16.72	15.95
pcb-layout	34.71	9.81	9.73
ppt-doc	23.17	12.14	9.06
twist-tunnel	37.72	28.55	19.02
video-conferencing	39.86	20.21	17.12
Average	35.62	16.99	13.91
BirdsInCage	0.04	0.08	0.01
CrowdRun	0.00	0.00	0.00
Kimono	0.00	0.01	0.00
ParkScene	0.00	0.00	0.00
Traffic	0.10	0.14	0.03
Average	0.02	0.03	0.01

Table 3.5: Joint probability ($P_i = P(A_i, CA_i, T)$) that assumption A_i is valid and conditions CA_i apply in a given window. $P_{1,2}$ provides the joint probability that both assumptions are valid and there conditions apply ($P(A_1, A_2, CA_1, CA_2, T)$)

four equations, this needs around 86 arithmetic operations. Note that the equations should only be resolved when the conditions for assumption 1 are present (i.e. that there are at most four unique luma values in the current window).

When for example a gradient texture is used (e.g. more unique colors are present in the sliding window), this assumption fails to resolve to a solution and the output of this assumption will be ignored. Therefore, the next assumption is introduced to provide a solution that will cover these use cases.

3.5.2.2 Assumption 2

The second assumption A_2 assumes that if the luma component has a gradient texture, the chroma component has the same (or the inverse) texture as the luma component between the minimum and maximum value. When there is a similar texture between the components, the chroma component would increase if the luma component increases. The chroma component decreases when the luma component increases when they have the inverse texture. The conditions for the second assumption (CA_2) require that the luma component has a gradient texture (e.g. continuously increasing or decreasing luma values in the current window). Table 3.4 provides the probability $P(CA_2|T)$ that the luma component has a gradient texture and the probability $P(A_2|CA_2, T)$ that the chroma component also has a gradient texture if the luma has a gradient texture. Also in this assumption there is a high probability of 91.73% for screen content that the assumption is true. Representing on average 16.99% of the tested windows (see Table 3.5). For the reader's convenience a worked out example (see Figure 3.12) is provided to illustrate how this assumption can reconstruct the chroma component. In this example,

$$\boldsymbol{L} = \begin{bmatrix} 80 & 80 & 80 \\ 80 & 75 & 75 \\ 80 & 70 & 70 \\ 80 & 65 & 65 \\ 80 & 60 & 60 \\ 80 & 60 & 50 \end{bmatrix}$$

$$\boldsymbol{C} = \begin{bmatrix} 100 & 115 \\ 100 & 134 \end{bmatrix}$$

$$\boldsymbol{R} = \begin{bmatrix} 100 & 100 & 100 \\ 100 & 110 & 110 \\ 100 & 120 & 120 \\ 100 & 130 & 130 \\ 100 & 140 & 140 \\ 100 & 140 & 160 \end{bmatrix}$$

Figure 3.12: Example where the original chroma component $\boldsymbol{R}$ has the inverse texture as the luma component $\boldsymbol{L}$. $\boldsymbol{C}$ is the sub-sampled version of $\boldsymbol{R}$.

the original chroma component $\boldsymbol{R}$ has the inverse texture as the luma component $\boldsymbol{L}$. The first assumption cannot be used as there are not enough pixels with equal luma values to solve equations (3.12) to (3.15).

To analyze and validate this assumption, first a sub-sampled version $\boldsymbol{D}$ of the luma component $\boldsymbol{L}$ is calculated. New luma pixel values are calculated in a similar fashion as the sub-sample filter used in the encoder:

$$\begin{aligned} d_{0,0} &= \frac{-l_{0,0} + 9l_{1,0} + 9l_{2,0} - l_{3,0}}{16} \\ d_{0,1} &= \frac{-l_{0,2} + 9l_{1,2} + 9l_{2,2} - l_{3,2}}{16} \\ d_{1,0} &= \frac{-l_{2,0} + 9l_{3,0} + 9l_{4,0} - l_{5,0}}{16} \\ d_{1,1} &= \frac{-l_{2,2} + 9l_{3,2} + 9l_{4,2} - l_{5,2}}{16} \end{aligned} \tag{3.16}$$

$$\boldsymbol{D} = \begin{bmatrix} d_{0,0} & d_{0,1} \\ d_{1,0} & d_{1,1} \end{bmatrix} \quad \left(= \begin{bmatrix} 80 & 72.5 \\ 80 & 68.8125 \end{bmatrix} \right). \tag{3.17}$$

The result of calculating $\boldsymbol{D}$ for the example in Figure 3.12 is provided between brackets. The sub-sampled luma component $\boldsymbol{D}$ is then normalized $\boldsymbol{N_D}$ between

its minimum and maximum. When there is no texture in $\boldsymbol{D}$ ($\max(\boldsymbol{D}) = \min(\boldsymbol{D})$), this assumption can not be used.

$$\boldsymbol{N_D} = \frac{\boldsymbol{D} - \min(\boldsymbol{D})}{\max(\boldsymbol{D}) - \min(\boldsymbol{D})} \quad \left(= \begin{bmatrix} 1 & 0.564 \\ 1 & 0 \end{bmatrix} \right) \tag{3.18}$$

Both $\boldsymbol{D}$ and $\boldsymbol{N_D}$ are calculated as floating points so there are no rounding errors. However the chroma component $\boldsymbol{C}$ has integer values with rounding errors $\delta_{i,j}$. Therefore both extrema will be calculated for $\boldsymbol{N_C}$ taking into account the possible rounding errors:

$$\boldsymbol{N_{C_{min}}} = \frac{\boldsymbol{C} - \min(\boldsymbol{C}) - 1}{\max(\boldsymbol{C}) - \min(\boldsymbol{C}) + 1} \quad \left(= \begin{bmatrix} -0.029 & 0.400 \\ -0.029 & 0.943 \end{bmatrix} \right) \tag{3.19}$$

$$\boldsymbol{N_{C_{max}}} = \frac{\boldsymbol{C} - \min(\boldsymbol{C}) + 1}{\max(\boldsymbol{C}) - \min(\boldsymbol{C}) - 1} \quad \left(= \begin{bmatrix} 0.030 & 0.485 \\ 0.030 & 1.061 \end{bmatrix} \right) \tag{3.20}$$

This assumption is considered valid if all elements from the normalized sub-sampled luma component are in the range of the elements of the normalized sub-sampled chroma component (Note that this is not true for the given example):

$$\boldsymbol{N_{C_{min_{i,j}}}} \leq \boldsymbol{N_{D_{i,j}}} \leq \boldsymbol{N_{C_{max_{i,j}}}} \quad (i,j) \in D. \tag{3.21}$$

This only holds true when the texture in both components is similar. The (full-resolution) luma component is normalized with the minimum and maximum of the sub-sampled luma component and scaled to the sub-sampled chroma component to reconstruct the up-sampled chroma component $\boldsymbol{R'}$:

$$\boldsymbol{N_L} = \frac{\boldsymbol{L} - \min(\boldsymbol{D})}{\max(\boldsymbol{D}) - \min(\boldsymbol{D})} \tag{3.22}$$

$$\boldsymbol{R'} = (\boldsymbol{N_L}.(\max(\boldsymbol{C}) - \min(\boldsymbol{C}))) + \min(\boldsymbol{C}). \tag{3.23}$$

If the normalized sub-sampled luma component did not fit the range of the normalized sub-sampled chroma component (equation 3.21), the inverse normalization $\boldsymbol{N'_D}$ is also tested:

$$\boldsymbol{N'_D} = \frac{\max(\boldsymbol{D}) - \boldsymbol{D}}{\max(\boldsymbol{D}) - \min(\boldsymbol{D})} \quad \left(= \begin{bmatrix} 0 & 0.436 \\ 0 & 1 \end{bmatrix} \right) \tag{3.24}$$

Like in equation 3.21 the inverse normalized sub-sampled luma component should element wise be in the range of the normalized sub-sampled chroma component. If this is true (like in the example) it is assumed that the chroma and luma component share the inverse texture.

$$\boldsymbol{N}_{\boldsymbol{C}min_{i,j}} \leq \boldsymbol{N}'_{\boldsymbol{D}_{i,j}} \leq \boldsymbol{N}_{\boldsymbol{C}max_{i,j}} \quad (i,j) \in D. \tag{3.25}$$

Then, the luma texture is inverse transferred to the up-sampled chroma component.

$$\boldsymbol{R'} = \max(\boldsymbol{C}) - (\boldsymbol{N_L}.(\max(\boldsymbol{C}) - \min(\boldsymbol{C}))) \tag{3.26}$$

$$\left(\boldsymbol{N_L} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0.709 & 0.709 \\ 1 & 0.418 & 0.418 \\ 1 & 0.127 & 0.127 \\ 1 & -0.164 & -0.164 \\ 1 & -0.164 & -0.745 \end{bmatrix} \quad \boldsymbol{R'} = \begin{bmatrix} 100 & 100 & 100 \\ 100 & 109.89 & 109.89 \\ 100 & 119.78 & 119.78 \\ 100 & 129.67 & 129.67 \\ 100 & 139.56 & 139.56 \\ 100 & 139.56 & 159.35 \end{bmatrix} \right). \tag{3.27}$$

If, in the example, $\boldsymbol{R'}$ is rounded, all pixels except $r_{5,2}$ have the same reconstructed value as in the input $\boldsymbol{R}$. This deviation (159 instead of 160) is caused by the rounding errors in $\boldsymbol{C}$. More important is the good texture reconstruction of the chroma component with the guidance of the luma component. The second assumption needs on average 83 arithmetic operations when the conditions for this assumption $P(CA_2|T)$ are true. As it is expected that components with a similar (linear transformed) gradient texture provide the same normalized sub-sampled component, equations 3.21 and 3.25 will detect if the normalized sub-sampled chroma component have a similar structure with the respectively normalized or inverse normalized sub-sampled luma component.

The probability that the conditions for the two assumptions $P(CA_1, CA_2|T)$ apply are shown in Table 3.4. On average $30.09\ \%$ of the windows in screen content sequences are tested with both assumptions. Which means that $43.20\ \%$ and $7.33\ \%$ of the windows are only testing respectively assumption 1 and assumption 2. As the proposed solution has a linear time complexity O(n) regarding the resolution of the video stream. The algorithm can be parallelized on the IPU to provide a real-time implementation if needed.

3.5.3 Results

The proposed method is verified with different sequences used in the common test conditions of the HEVC range extensions. Two groups of test sequences are used, the first group are the screen content sequences: cad-waveform, pcb-layout, ppt-doc, twist-tunnel and video-conference. The second group contains natural camera captured content: CrowdRun, Traffic, BirdsInCage, Kimono and ParkScene. The original sequences are Y'CbCr 4:4:4 chroma subsampling sequences. For lossy compression the HM 16.4 reference codec[1] with the Super High Tier (SHT) con-

[1]The newest version at the time of writing is HM 16.12. No significant difference is expected between these two versions.

figuration with QP values 12, 17, 22 and 27 is used. There is no added value to use Main tier QP configurations as the quality degradation from the compression artifacts is not acceptable in the professional applications considered the scope in this chapter. The improvements in the chroma components are objectively measured using PSNR and Structural SIMilarity (SSIM) [57]. However, objective quality improvements are not always a clear indication for screen content. This is mainly because the PSNR is based on the Mean Square Error (MSE) of the full frame, while screen content artifacts are typically located around the sharp edges, which may be not homogeneously distributed over the entire frame. As a result, subjective quality gain can sometimes even be identified with small PSNR gains. SSIM tries to estimate the perceived error by assuming a strong dependency between pixels that are spatially close. It will, in this use case, provide a better objective quality metric. The improvements where subjectively verified and some visual examples will be provided.

3.5.3.1 Lossless compression

	PSNR-Chroma [dB]		
Sequence	Anchor	Proposal	Gain
cad-waveform	23.07	26.45	3.38
pcb-layout	26.26	28.76	2.50
ppt-doc	31.24	31.99	0.74
twist-tunnel	36.46	37.37	0.91
video-conferencing	29.85	30.50	0.64
Average			1.63
BirdsInCage	39.78	39.25	-0.53
CrowdRun	37.25	37.25	-0.00
Kimono	40.96	40.88	-0.08
ParkScene	38.30	38.29	-0.01
Traffic	42.65	42.02	-0.62
Average			-0.25

Table 3.6: PSNR-Chroma [dB] improvement with guided chroma reconstruction for uncompressed content.

Table 3.6 provides the PSNR-chroma results for lossless compressed Y'CbCr 4:2:0 chroma subsampling bitstreams using standard HEVC/H.265 proposed chroma up-sampling as anchor. For screen content, the improvements are in the range of 0.64 dB for the video-conferencing sequence up to 3.38 dB improvement for the cad-waveform sequence, with an average of 1.63 dB over all test sequences. As the screen content sequences contain many details at the resolution of a single pixels (e.g. text, one pixel width lines, etc.) the PSNR is an order of magnitude

	SSIM-Chroma		
Sequence	Anchor	Proposal	Gain
cad-waveform	0.794	0.921	0.127
pcb-layout	0.845	0.923	0.078
ppt-doc	0.911	0.927	0.016
twist-tunnel	0.984	0.988	0.004
video-conferencing	0.877	0.909	0.032
Average			0.051
BirdsInCage	0.932	0.927	-0.005
CrowdRun	0.910	0.910	0.000
Kimono	0.939	0.938	0.000
ParkScene	0.917	0.917	0.000
Traffic	0.973	0.971	-0.003
Average			-0.002

Table 3.7: SSIM-Chroma improvement with guided chroma reconstruction for uncompressed content.

lower for this sequence set compared to the natural content sequences. However the visual quality of the screen content with this low PSNR values is still good (see Figure 3.13), except for the details around sharp edges. The SSIM results for the same configuration can be found in Table 3.7. These results show the same trend as the PSNR results. The cad-waveform sequence provides the best SSIM-chroma gain of 0.127, while an average SSIM-chroma gain of 0.051 is measured for the screen content sequences. While the sequences cad-waveform, pcb-layout, ppt-doc have very diverse SSIM results in the anchor version, in the proposed method they all provide a SSIM around 0.925.

To subjectively verify the proposed solution, a detail of the cad-waveform sequence is provided in Figure 3.13. Most colors of the single pixel wide lines can be reconstructed with the proposed method. Figure 3.16 demonstrates perfect reconstruction of a detail of the pcb-layout sequence, while the anchor provides a smoothed output without detail.

For natural content, the results range from 0.00 dB loss for the CrowdRun sequence to a loss of 0.62 dB for the Traffic sequence. The average loss of SSIM-chroma for the natural content sequences is 0.002. For the tested sequences this loss is rather negligible, so there is no need for algorithms that classify content to decide whether to activate the proposed up-sampling filter or not, which is a considerable advantage.

3.5.3.2 Lossy compression

Results for experiments where HEVC/H.265 compression is used can be found in Table 3.8. There is always a PSNR-chroma gain measured for all QP values. The

	PSNR-Chroma [dB]											
	QP=12			QP=17			QP=22			QP=27		
Sequence	Anchor	Proposal	Gain	Anchor	Proposal	Gain	Anchor	Proposal	Gain	Anchor	Proposal	Gain
cad-waveform	23.07	25.65	2.58	23.07	25.16	2.09	23.06	24.78	1.71	23.05	24.20	1.15
pcb-layout	26.27	27.50	1.23	26.26	27.20	0.94	26.23	26.87	0.64	26.21	26.62	0.41
ppt-doc	31.25	31.63	0.38	31.23	31.52	0.29	31.22	31.47	0.25	31.12	31.30	0.18
twist-tunnel	36.49	37.03	0.54	36.47	36.88	0.41	36.43	36.78	0.35	36.28	36.53	0.25
video-conferencing	29.85	30.34	0.48	29.83	30.21	0.39	29.80	30.11	0.31	29.72	29.96	0.24
Average			1.14			0.81			0.65			0.45
BirdsInCage	39.78	39.31	-0.47	39.67	39.31	-0.36	39.46	39.23	-0.23	39.15	38.95	-0.20
CrowdRun	37.09	37.08	-0.00	36.72	36.69	-0.03	35.95	35.91	-0.04	34.87	34.85	-0.02
Kimono	40.48	40.18	-0.29	39.77	39.14	-0.62	39.00	38.74	-0.26	38.23	38.10	-0.13
ParkScene	38.06	38.02	-0.04	37.62	37.15	-0.46	36.83	36.64	-0.18	35.97	35.93	-0.04
Traffic	41.84	41.40	-0.44	41.05	40.79	-0.26	40.12	39.99	-0.14	38.95	38.86	-0.09
Average			-0.25			-0.36			-0.17			-0.10

Table 3.8: PSNR-Chroma [dB] improvements with guided chroma reconstruction for lossy HEVC/H.265 compressed content.

	SSIM-Chroma											
	QP=12			QP=17			QP=22			QP=27		
Sequence	Anchor	Proposal	Gain	Anchor	Proposal	Gain	Anchor	Proposal	Gain	Anchor	Proposal	Gain
cad-waveform	0.794	0.906	0.112	0.793	0.891	0.098	0.792	0.874	0.082	0.788	0.848	0.059
pcb-layout	0.845	0.900	0.054	0.844	0.880	0.035	0.842	0.871	0.029	0.840	0.863	0.024
ppt-doc	0.911	0.921	0.009	0.911	0.918	0.007	0.910	0.915	0.005	0.906	0.909	0.003
twist-tunnel	0.985	0.987	0.003	0.985	0.987	0.003	0.984	0.986	0.002	0.983	0.985	0.001
video-conferencing	0.877	0.902	0.025	0.875	0.894	0.019	0.873	0.888	0.015	0.868	0.880	0.012
Average			0.048			0.032			0.027			0.020
BirdsInCage	0.930	0.923	-0.006	0.927	0.922	-0.005	0.923	0.920	-0.003	0.920	0.917	-0.002
CrowdRun	0.906	0.906	0.000	0.896	0.896	0.000	0.875	0.875	-0.001	0.851	0.851	-0.001
Kimono	0.933	0.931	-0.002	0.922	0.917	-0.006	0.911	0.908	-0.003	0.906	0.904	-0.002
ParkScene	0.912	0.911	0.000	0.901	0.896	-0.005	0.881	0.878	-0.003	0.863	0.862	-0.001
Traffic	0.966	0.964	-0.003	0.959	0.957	-0.002	0.951	0.949	-0.001	0.940	0.939	-0.001
Average			-0.002			-0.004			-0.002			-0.001

Table 3.9: SSIM-Chroma improvements with guided chroma reconstruction for lossy HEVC/H.265 compressed content.

best sequence is, like in the uncompressed case, the cad-waveform sequence. This sequence has a PSNR-chroma gain from 2.58 dB at $QP = 12$ down to 1.15 dB at $QP = 27$. The ppt-doc sequence has the lowest gain of 0.18 dB at $QP = 27$. On average there is a gain from 1.14 dB at $QP = 12$ down to 0.45 dB at $QP = 27$. As also can be seen on the lossless use case, the sequences with the lowest SSIM-chroma get the most SSIM-chroma improvement. The cad-waveform sequence is improved from 0.794 to 0.906, while the pcb-layout sequence is improved from 0.845 to 0.900. The proposed method can improve less quality when the encoder is using a higher QP value (e.g. higher compression ratio). This can be explained as higher QP values will introduce more encoding artifacts on the luma and chroma component. In the first assumption this can cause more unique colors in a window so the conditions for the assumption does not apply any more. In the next assumption only a rounding error is expected in the chroma component. However if there are additional encoding artifacts, it is less probable that similar texture will be detected.

The camera captured sequences have an average PSNR-chroma loss of 0.36 dB at $QP = 17$ down to 0.010 dB at $QP = 27$. As the guided chroma reconstruction algorithm is not designed for this content the gain is not predictable over different QP values. The Kimono sequence has its highest PSNR-chroma loss of 0.62 dB at $QP = 17$ while it has only 0.13 dB loss at $QP = 27$.

A PSNR-Chroma comparison with other methods for chroma up-sampling with screen content sequences is provided in Figure 3.18. The proposed method of guided chroma reconstruction (GCR) is compared with: nearest neighbor, bilinear interpolation, our implementation of the method introduced by Itoh et al. [51], guided image filter with fixed radius and the anchor method (4-tabs chroma filter proposed in HEVC). The proposed method continues to improve the chroma component at higher bitrates, when there is no noticeable improvement anymore in the other methods. This can be explained by the fact that the guided chroma reconstruction method is still using improvements in the luma component which it uses as a guide. Around 7Mbps the proposed method also has achieved the maximum quality improvement for the tested screen content sequences.

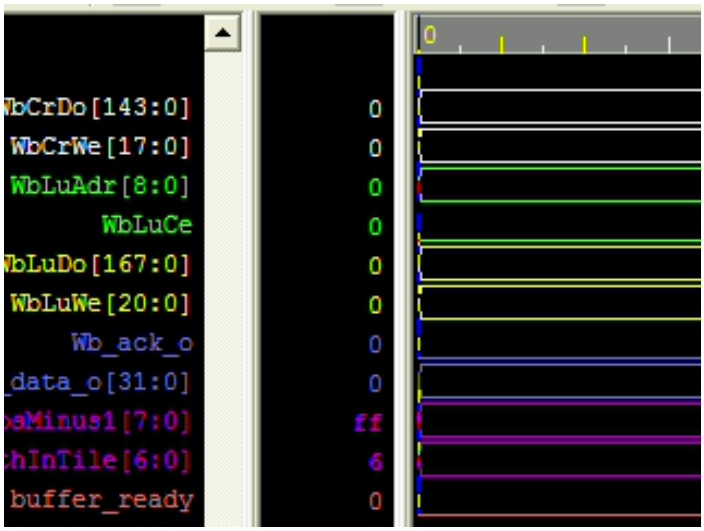
To illustrate the overall performance of the proposed solution an RD-curve comparison of the proposed solution is made with Y'CbCr 4:4:4 chroma subsampling based compression (see Figure 3.19). Also in this experiment, the HM-16.4 reference implementation is configured with lowdelay main (P-frames only) configuration. The graph provides results averaged over all screen content sequences. In the previous RD-curves all streams had the same luma component so the PSNR-Chroma was used in the comparison. In this case the streams have a different luma component so the combined PSNR [58] is used. This is a weighted sum of the PSNR per component:

$$\text{PSNR} = (6 \cdot \text{PSNR}_{Y'} + \text{PSNR}_{Cb} + \text{PSNR}_{Cr})/8 \tag{3.28}$$

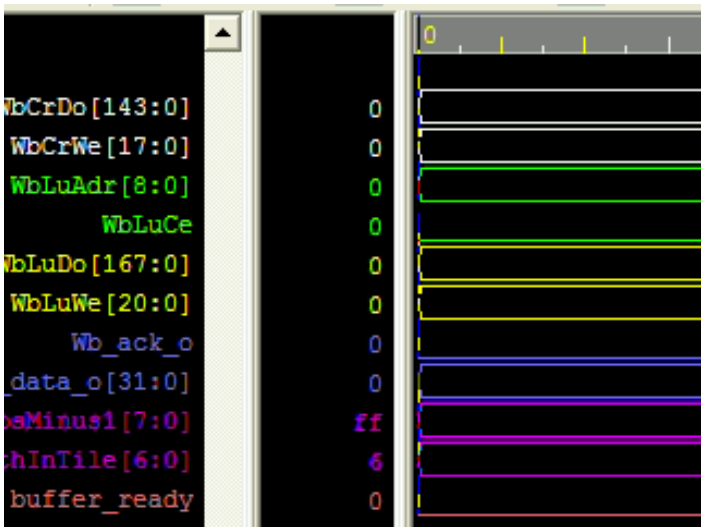
At lower bitrates Y'CbCr 4:2:0 chroma subsampling has better performance as the encoder artifacts have more influence on quality as the sub-sample artifacts. This is different at higher bandwidths, the Y'CbCr 4:4:4 chroma subsampling implementation can keep improving the chroma components while the Y'CbCr 4:2:0 chroma subsampling are limited to their ability to reconstruct the chroma component. It should however be noted that in the given use case in the introduction, using Y'CbCr 4:4:4 chroma subsampling has a high impact on the system design. Simulcasting or transcoding could be used to stream a Y'CbCr 4:4:4 chroma subsampling stream to professional decoders and Y'CbCr 4:2:0 chroma subsampling stream to consumer electronics. Next to the additional codec implementations also the control layer of the system should be extended to support this.

3.5.4 Conclusion

The goal of the work presented in this section is to achieve better quality for screen content coding, even if an unmodified encoder is used. It is shown that by using guided chroma reconstruction, the quality is increased for both lossless and lossy use cases. Two assumptions have been investigated to determine the effect of the chroma sub-sampling filter. The first assumes an equality of the chroma pixels when the luma pixels have the same value. The second evaluates that the chroma component has the same texture as the luma component. The assumptions have to be validated first before the output of the guided chroma reconstruction can be used. If both assumptions are not valid, a common up-sampling filter is used. The impact for this technique on non-screen content is limited. For screen content coding an average PSNR-chroma gain of 1.63 dB and 0.051 SSIM-chroma improvement is measured, while for natural camera captured content a loss of 0.25 dB is measured.



(a)



(b)

Figure 3.13: Detail from the cad-waveform test sequence. The proposed method (b) with 26.45 dB provides sharper edges and more original colors compared to the anchor method (a) with 23.07 dB.

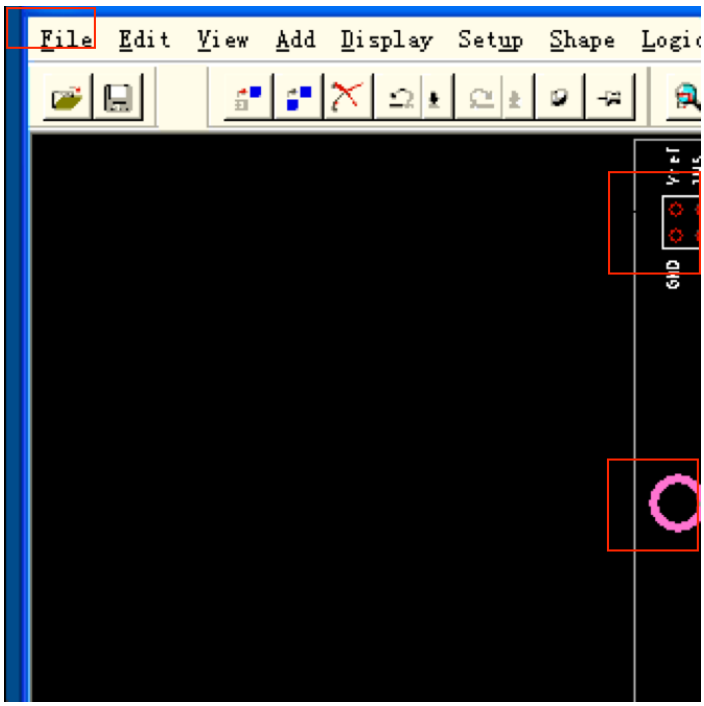
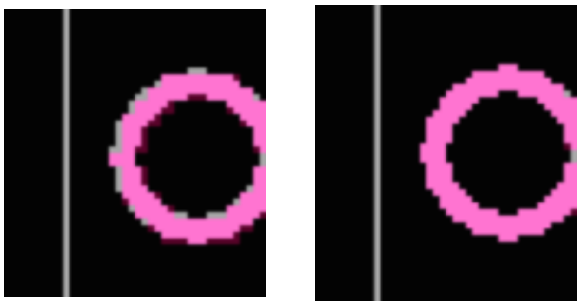


Figure 3.14: Fragment from the PCB layout sequence showing the positions of the details in next figures.

(a) (b)

Figure 3.15: Detail from a small red circle on black background from the pcb-layout test-sequence (Figure 3.14). While the anchor method (a) provides a blurred and distorted circle, the guided chroma reconstruction method provides a perfect reconstruction (b).



(a) (b)

Figure 3.16: Detail of a pink circle from the pcb-layout test-sequence (Figure 3.14). There is a chroma offset between the chroma and luma circle in the anchor method (a). The proposed method (b) aligns both the chroma and luma circle.

(a) (b)

Figure 3.17: Detail of the taskbar in the pcb-layout test-sequence (see Figure 3.14). Some additional colors are introduced with chroma subsampling (a) and are removed with guided chroma reconstruction (b).

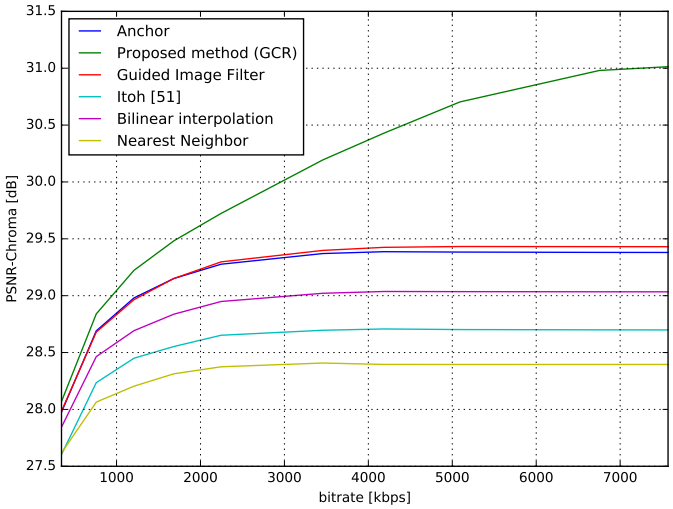


Figure 3.18: Average RD curve for screen content sequences when different chroma up-sampling methods are used.

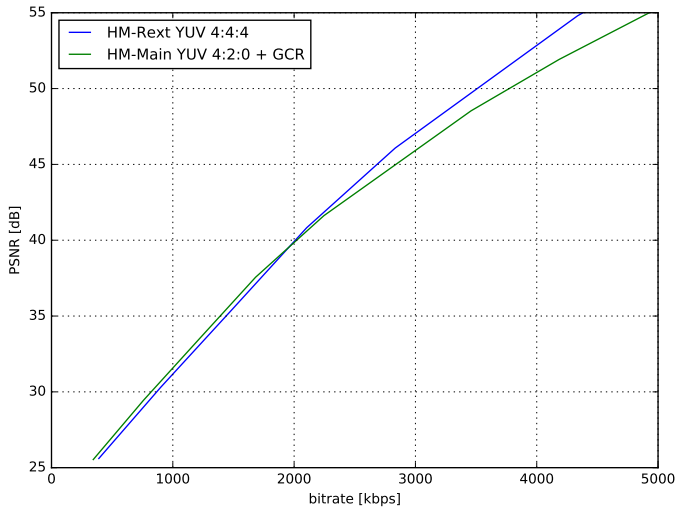


Figure 3.19: Average RD curve for screen content sequences with HEVC HM Main profile (Y'CbCr 4:2:0 chroma subsampling) with proposed solution and encoding with proposed method and HM main-Rext profile (Y'CbCr 4:4:4 chroma subsampling)

Proposed Method	PSNR-Chroma Gain	Additional Bandwidth
Optimal radius *	1.91	High
Histogram analysis *	0.81	No
MDO *	1.57	Medium
GCR	1.63	No

*Table 3.10: Overview of proposed methods for chroma reconstruction of screen content after Y'CbCr 4:2:0 chroma subsampling. Methods marked with * are using the guided image filter.*

3.6 Conclusions and original contributions

The goal of the research presented in this chapter is to explore how H.265/HEVC Main profile can be used in a professional video system. This with the goal to improve the interoperability with consumer electronics. It is identified that the quality is not good enough for screen content. The quality introduced by chroma subsampling introduces unwanted artifacts. This is especially true around sharp edges text and pixel-size details. Chroma subsampling cannot be avoided as Y'CbCr 4:2:0 chroma subsampling is the only supported chroma subsampling format in H.265/HEVC Main profile.

Different methods are proposed to enhance the chroma components and reduce the artifacts introduced by chroma subsampling. While designing solutions for this problem it is taken into account that there are additional industrial requirements. Proposed solutions should be cost effective and the solutions should be standards compliant to guarantee interoperability with third party devices, especially consumer electronics.

Table 3.10 provides an overview of the different proposed methods in this chapter. In a first approach, the guided image filter is used to enhance the chroma components of the decoded stream. Using the guided image filter with a fixed parameter configuration does not improve the quality of screen content. Improvements of the content are very local and a more adaptive approach is explored. If the *optimal radius* parameter is selected for each pixel, there is a significant quality improvement. However, for signaling the filter configuration from the encoder to the decoder additional bandwidth is required. To reduce the extra bandwidth, different techniques are explored to reconfigure the guided image filter.

A prediction model is created based on *histogram analysis* of the luma component. This prediction model is used to configure the radius parameter of the guided image filter. This method works well for screen content with a limited amount of unique colors. The model does not work if more unique colors are used (e.g. gradient or transparent areas), it even introduces new artifacts in this case. This is caused because there is not much similarity between the optimal radius parameter

of neighboring pixels in areas with lots of unique colors. This makes it difficult to find a better prediction model.

Therefore, it is opted to extend the bitstream with additional information so the decoder can reconfigure the guided image filter. This information will be ignored by third party decoders making the solution standards compliant. With the *Minimum difference offset* (MDO) technique, the encoder only signals to use the guided image filter if there is a large improvement in the output of the filter. This method prevents the introduction of new artifacts. By configuring the MDO, the user also has the possibility to balance between quality and additional bandwidth. However, this solution needs modifications to both the encoder and the decoder.

With the last proposed solution, *Guided Chroma Reconstruction* (GCR), the decoder can also enhance the screen content stream from third party devices (including consumer electronics). This method does not require any additional information from the encoder. This method is, like previous experiments, based on the assumption that for screen content the luma and chroma component have correlation, additionally it also takes the chroma subsample filter into account. Guided chroma reconstruction only adapts a pixel component if there is a high probability that the assumptions are correct. This method provides improvements for both lossless and lossy encoding. The impact on camera captured content is very low as it detects little correlation between the components and therefore no significant changes are done to the chroma component. As a consequence, this method can also be enabled to reduce the system complexity.

In future work, guided chroma reconstruction can be extended to verify that this method also works with other chroma subsampling filters. As the chroma subsample filter normally does not change in the same video stream, it should also be possible to automatically detect the filter that the decoder was using for chroma subsampling. For example, if different chroma subsample filters are used to estimate the probability that there is cross component correlation. The filter with the highest probability score will most likely be used in the chroma subsampling process. If all filters report a low probability of correlation, the used filter is not tested or the content does not have screen content. The disadvantage of testing for more chroma subsampling filters is that it computationally complex and it would take us out of the scope of this work.

The work on chroma reconstruction has led to the following publications:

- T. Vermeir, J. Slowack, S. Van Leuven, G. Van Wallendael, J. De Cock and R. Van de Walle, *Adaptive guided image filtering for screen content coding*, 2014 IEEE International Conference on Image Processing (ICIP), Paris, 2014, pp. 5561-5565.
- T. Vermeir, J. Slowack, R. Van Belle, S. Van Leuven, G. Van Wallendael, J. De Cock, R. Van de Walle, *Guided Chroma Reconstruction for Screen Content Coding*, in IEEE Transactions on Circuits and Systems for Video Technology. Accepted for future publication.

The following contributions where made to the JCT-VC and MPEG standardization process:

- T. Vermeir, *Use cases and requirements for lossless and screen content coding*, JCTVC-M0172, Incheon, Korea, April 2013.
- T. Vermeir, J. Slowack, J. De Cock, G. Van Wallendael, S. Van Leuven, *AhG8: Guided Image Filtering for Screen Content Coding*, JCTVC-N0148, Vienna, Austria, July 2013.

The following patents were filed:

- T. Vermeir, US 20150016720, *Guided image filtering for image content*, publication date: 15/01/2015
- T. Vermeir, WO 2016113396, *Method and apparatus for chroma reconstruction*, publication date: 21/07/2016

4

Complexity Constrained Encoding

4.1 Introduction

There are different challenges when using H.265/HEVC Main profile in professional video systems. In the previous chapter, problems related to Y'CbCr 4:2:0 chroma subsampling for screen content coding where handled. It was assumed that the encoder and/or decoder was modified to enhance the reconstructed chroma components. However not mandatory, it was assumed that hardware acceleration was available for encoding and decoding of the H.265/HEVC Main profile stream. In this chapter, it will be assumed that hardware acceleration is not available and the video stream should be encoded in software.

When encoding in software, the encoding application itself is usually not the only software process running on the host machine. Other processes can run in parallel, including operating system processes, and even other encoders / decoders. For example, this happens when SCADA content is generated on the same computer as the encoding process (see Section 1.2.2). Another example could be the screen sharing use case in the collaboration division (see Section 1.2.3). When the load on the host machine becomes significant, the software processes may fight for the same resources and as such, influence each others processing speed. This can be problematic, particularly for a video encoder that is expected to deliver output at regular time intervals. Although buffering may solve some of these issues, adding a buffer introduces additional delay that may not be desired in a low-delay video application. Therefore, in many cases, the solution consists of carefully fine tuning

configuration parameters and/or over provisioning resources (e.g., selecting one of the more high-end cloud instances). This results in additional effort and financial cost.

Furthermore, when over dimensioning the system for the statical worst case (e.g., the maximum number of video streams the cloud service should be able to handle), the configuration may be suboptimal for other cases since in these other cases system resources may not be fully used. When more resources are available, a video may be able to improve quality further, for example. So instead of dimensioning a system for the worst case, it would be interesting to assign a complexity budget to each software process, and have the process reconfigure itself to respect this budget. Such an automatic reconfiguration is particularly useful in cases where the load on the host changes dynamically and/or the same software is expected to flexibly run on multiple hardware platforms (e.g., cloud instances of different providers).

In this chapter, we will explore techniques for complexity control in the context of H.265/HEVC [59] video encoding, a computationally complex process. According to [60], by adding new compression tools as well as extending already supported compression tools, HEVC encoding consumes 9 to 502 % more computational complexity compared to its predecessor H.264/AVC [61], depending on the configuration settings. Note that the actual computational complexity also depends on the complexity of the video sequence being encoded which further motivates the need for complexity control.

The state-of-the-art in complexity optimization for HEVC is extensive. However, most algorithms described in the literature do not provide complexity control, but instead focus only on reducing complexity, typically, through early termination of the mode decision or motion search process. For example, Choi et al. [62] proposed to not evaluate split when the best mode is SKIP. Jaehwan et al. [63] reduce complexity with 35% by deciding early in the encoder evaluation process whether or not to use SKIP mode. Miok et al. [64] propose to early terminate motion search by estimating the RD cost of the merge mode based on the results obtained for motion vector prediction. Zhaoqing et al. [65] propose to use a technique for early merge mode decision based on information from the all-zero block and motion estimation, exploiting also correlation between different CU depths to reduce complexity.

A limited number of researchers have focused explicitly on complexity control, often labeling their work as *complexity constrained encoding* or *complexity scalable encoding*. For example, Kannangara et al. [66] propose techniques for frame-level complexity control of a real-time H.264/AVC encoder. The complexity is controlled by choosing between fast encoding using SKIP mode and full evaluation, based on an estimation of the Lagrangian rate-distortion-complexity (RDC) cost for both options. Correa et al. [67] propose a complexity scalability

method for HEVC in which CTUs are dynamically constrained by limiting the prediction block (PB) and setting a maximum tree depth for each CTU. Zhao et al. [68] propose a hierarchical complexity allocation scheme for HEVC based on linear programming. While these methods can provide good complexity control, they report relative high variation of the encoding time over multiple GOPs, which make them less suitable for a low latency real-time video encoder.

4.2 Measuring complexity

A lot of research has already been done on the complexity of video encoding, mostly in the context of complexity reduction. However, there is no widely accepted definition for complexity. In most research projects, the complexity definition and complexity metric is adapted to a specific use case. For example, the following complexity definitions are used to define the complexity of a video encoding implementation: code analysis, power consumption, software profiling, runtime and others. Therefore, complexity measurements and improvements proposed in state-of-the-art should be compared critically as they might mean something different. For example, an algorithm optimized for reducing power consumption in a random access use case can potentially have increased encoding time in a low delay real-time use case.

4.2.1 Complexity definitions

Here are definitions of complexity used recently in state-of-the-art literature to describe the complexity of video encoding:

- **Code analysis**: When code analysis is used as a metric for complexity, the encoder implementation is split into atomic functions whose complexities are evaluated in terms of Millions of Operations executed Per Second (MOPS). Analysis of these instructions is done on a hypothetical Reduced Instruction Set Computer (RISC). This method is previously used in [69] and [70] to compare different implementations. This analysis focuses on the amount of operations per second and therefore does not take the delay of input-output operations into account. For example, it is assumed that the processor has an unlimited number of registers and unlimited memory access. This method of analysis works well to compare the complexity of different small parts of the encoder and to compare different (fast) implementations, because it removes the need to duplicate an exact copy of the testing environment. Typically when a fast encoding implementation is proposed, the only goal of the research is to prove that a novel algorithm produces the same (or similar) result as the old algorithm but with less instructions. When the difference is large enough it can be assumed that there is also an

improvement on real processors. While it might be possible to count the instructions of small parts of the encoder (e.g. DCT transform) manually, it is impractical to count the instructions that are needed to encode a whole frame. Another disadvantage of counting instructions is that parallelization (like WPP in H.265/HEVC) is not taken into account. As a conclusion, this method cannot be used easily to predict the complexity of a real-time video encoder implementation.

- **Power consumption**: Studies like [71], [72], [73] and [74] take the power consumption of the encoding device into account as a metric for the complexity. This is mainly interesting for mobile or battery powered devices. Typically these studies are interested in the power consumption of the whole device in a specific video transmission use case. The power consumption of the encoding device is not only caused by the algorithmic operations of the encoding process but also by other components like memory access and (wireless) network transmission. So, reducing the total power consumption does not automatically mean a faster encoding process or an encoding process with less algorithmic operations. For example, power consumption can be increased in the encoding algorithm to reduce the bandwidth and therefore decrease the power consumption on network transmission. While research in the power consumption of a video encoding process is useful and interesting, the focus in this chapter is constraining the encoding complexity to a complexity threshold.

- **Software profiling**: Software profiling is typically done by a lightweight process running parallel to the application (e.g. the video encoder) to be profiled. On a regular basis (e.g. elapsed time, instruction count or other interrupts) the profiler will take a snapshot of the current state of the encoding process. Afterwards, based on the profiling information and the binary file of the encoder application, the duration or instruction count for different functions of the encoder can be estimated. The resolution and reliability of the profiler is typically very different based on the profiler manufacturer, operating system and hardware architecture making the results very specific to the test environment. This method for complexity analysis is used in [75], [76] and [77]. Typically, the results of software profiling are only used to compare the complexity of different parts in the encoding process. As software profiling has influence on the encoding process this method cannot be used online to define the complexity of the full encoding process.

- **Runtime**: One of the most used metrics to define the complexity is to measure the runtime. This metric is also used in the development of H.265/HEVC Main profile [56]. Typically, the runtime of encoding all the frames of a sequence is compared between different implementations. However, this does

not take the variation of different frames into account. In most codec implementations there is a significant difference between the coding of I-, P- and B-frames. Therefore, some papers measure the complexity based on the encoding time of a GOP [78] or a single frame [67]. For real-time use cases measuring the encoding time of a GOP is only useful if the use case also allows latency of the duration of a GOP.

The final goal of the work in this chapter is to constrain the encoding process for real-time low latency streaming. To achieve real-time encoding (and streaming), the encoding time of every frame should be lower as a predefined time. Therefore, the encoding time of a single frame will be used as a the metric for complexity in this chapter. There are additional benefits of using the encoding time per frame as the complexity metric. It can be measured during encoding without significant overhead and the encoding time depends on the used hardware, the video sequence and possible optimizations in the codec. As a consequence, the runtime per frame is a good complexity metric in different configurations.

4.2.2 Real-time complexity measurement

The encoding time per frame will be used as the metric for complexity, measuring this value is not always trivial as for low values (i.e. in the case of real-time encoding) the accuracy of the measuring method should be taken into account. This section will explain that power-saving functionality should be disabled for reliable measurements of the encoding time.

Therefore, first some experiments will investigate the accuracy of measuring the encoding time per frame by using the system-wide real-time clock. These experiments are done on a Intel Xeon CPU E5-2620 [79] at 2.4 Ghz with a Ubuntu 14.04 [80] operating system and x265 1.8.

Figure 4.1 shows a heatmap of the measurements of the encoding time per frame T_m for the first 100 frames of the kimono sequence, the measurement is repeated 100 times. While horizontal lines where expected (i.e. same measurement in every iteration), spikes in T_m are measured randomly distributed over the frames and iterations.

$$T_m = T_a + \delta_s \tag{4.1}$$

This is caused because if T_m is measured, the measured value is the sum of the minimal encoding time T_a and a variable time introduced by the system δ_s. T_a contains the algorithmic complexity of the encoding process and the minimal overhead needed by the operating system to encode a frame. δ_s is the variable part. It is variable because the operating system is not real-time and therefore has non-deterministic behavior. Causes of variation in δ_s could be: cache misses,

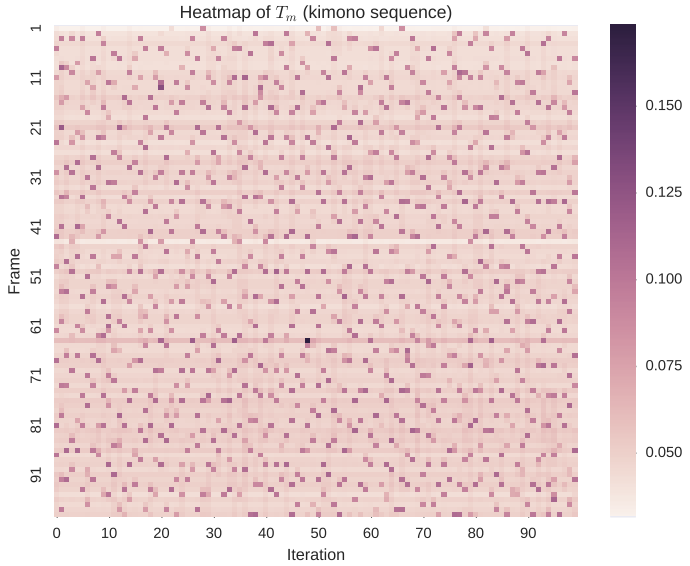


Figure 4.1: Heatmap of 100 measurements of the encoding time per frame T_m for the first 100 frames of the kimono sequence. Spikes in measurements of T_m are randomly distributed.

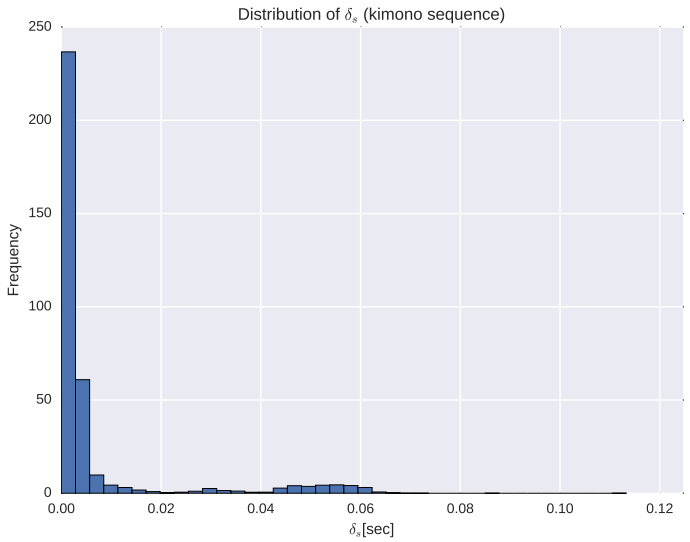


Figure 4.2: Distribution of δ_s for the kimono sequence. 10% of the measurements have a $\delta_s > 40$ msec.

memory access, or operating system interrupts. The complexity controller that will be proposed in Section 4.3.2 can only influence T_a. Therefore, high values of δ_s will decrease the accuracy of the complexity controller.

For measurement iteration i and frame n the encoding time $T_m^{n,i}$ can be measured. If there are enough measurements for a given frame, there should be a measurement j where $\delta_s^{n,j} = 0$ so $T_m^{n,j} = T_a^{n,j}$. Therefore:

$$T_a^n = \min_i T_m^{n,i} \tag{4.2}$$

$$\delta_s^{n,i} = T_m^{n,i} - T_a^n \tag{4.3}$$

The distribution of δ_s is visualized in Figure 4.2. This graph shows that 20% of the measurements have δ_s >4.5 msec and 10% of the measurements have δ_s >40 msec. For real-time encoding at 30 frames per second, the maximum encoding time per frame is 33 msec so the δ_s measured is unacceptably high.

The influence of different external factors (e.g. number of cores, CPU clock-speed, memory access) is investigated and the power-saving controller that is present (and default enabled) in any modern CPU has been identified as the component with the highest impact on δ_s of the measurements of encoding time. Modern CPUs try to optimize power usage by introducing power-saving modes [81] (also called C-states). Different C-states are defined ranging from Operating State (C0) where everything is fully operational to Deep Power Down (C6). In deeper C-states the CPU can stop internal components (e.g. clocks, floating point unit, memory cache) and reduce the internal voltage. The functionality of the CPU at different C-states is depending on the model of the CPU. The CPU will enter a deeper C-state if it is triggered by the operating system *idle* routine. Exact behavior and functionality is specific to the model of the CPU, but this is not noticeable from within the operating system. When the *wakeup* interrupt is sent, the CPU transitions from whatever C-state it is in to C0. This transition is an atomic operation and it takes more time to transition from deeper C-states, as some parts may need to power back on or reinitialize the internal state. In CPUs with multiple cores, there could also be some synchronization needed between the cores after C-state transitions. Transitions can generate a large number of Inter-Processor Interrupts (IPIs) and therefore can even lock CPU cores that are not transitioning from C-state. Via the operating system, the CPU can be configured to limit deeper C-states or even fully disable power-saving functionality for higher performance or better real-time behavior. This can be configured system wide or per application if needed. For example, to limit the maximum C-state system wide, the `processor.max_cstate` and `intel_idle.max_cstate` option can be added to the kernel line on a Linux system:

```
linux /boot/vmlinuz-3.16.0-44-generic ...
    processor.max_cstate=1 intel_idle.max_cstate=0
```

The system wide method is permanent (until the system is rebooted). If more dynamic control is required to have extremely low latency during certain hours, but more power savings at other times, there is a method to dynamically control which C-states are used. To dynamically control C-states on Linux, the file `/dev/cpu_dma_latency` should be opened and the maximum allowed latency should be written to it. This will prevent C-states with transition latencies higher than the specified value from being used, as long as the file is kept open.

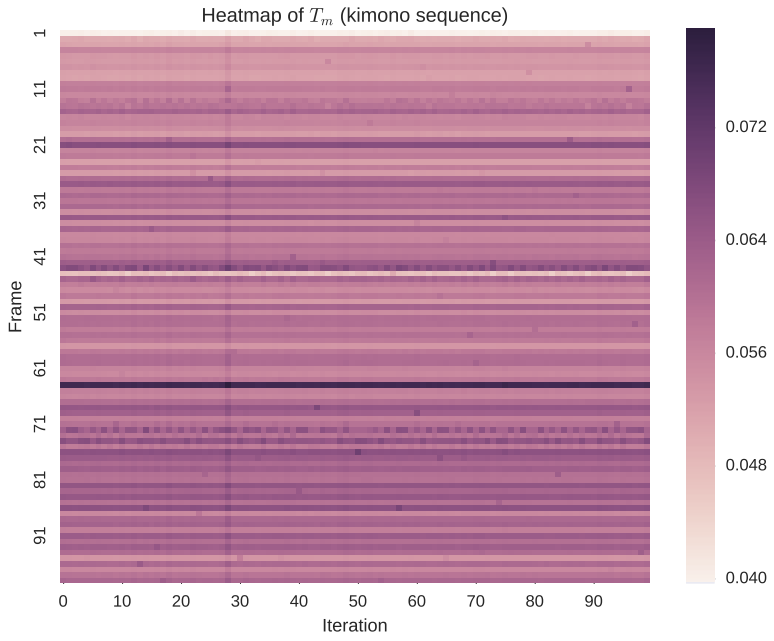


Figure 4.3: Heatmap of 100 measurements of T_m when power-saving is disabled for the first 100 frames of the kimono sequence. Horizontal lines in the heatmap provide visual feedback that T_m is similar for the same frame over multiple iterations.

In a new experiment with the same setup as in previous experiment, CPU power-saving was disabled. The results from these measurements are visualized in Figure 4.3 and Figure 4.4. There is a significant reduction in the δ_s, 90% of the measurements of δ_s are lower than 0.7 msec. Because the standard deviation (measured over multiple iterations) on T_a is 4.7 msec, δ_s measured by disabling power-saving modes and using the system real-time clock is low enough to build a complexity controller for a video encoder.

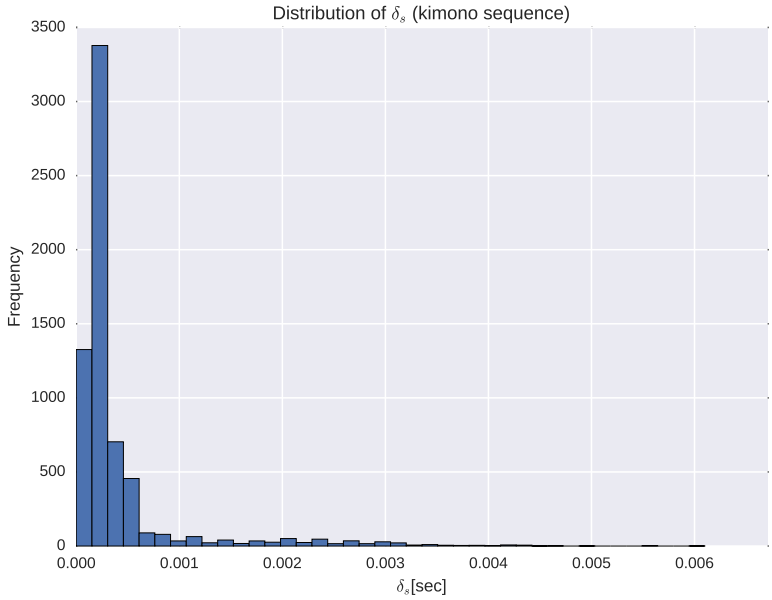


Figure 4.4: Distribution of δ_s when power-saving is disabled. 90% of the measurements $\delta_s < 0.7$ msec.

4.3 Defining a complexity threshold

Now it is clear how the complexity can be measured, it will be discussed in this section how the complexity is consumed in a H.265/HEVC Main profile software encoder. Most of the encoding time in the encoder is spent deciding the encoding modes for each CU. For an entire frame, there are M CUs (for each CTU, this will be maximum $1 + 2^2 + 2^4 + 2^8$ CU's). A particular CU at index m in this list ($m \in \{1, ..., M\}$) can be encoded using a number of different coding modes. However, CUs at a higher depth will only be evaluated when their parent CU does evaluate to split further. When the parent CU does not evaluate to split further, the CU is not added to the list of evaluated CUs. The length of the list of evaluated CUs is defined as Ω. For the n^{th} mode ($n \in \{1, ..., N\}$)[1] having distortion $D_{m,n}$ and rate $R_{m,n}$, we define the lagrangian RD cost $J_{m,n}$ as:

$$J_{m,n} = D_{m,n} + \lambda R_{m,n} \tag{4.4}$$

where λ is the Lagrange multiplier.

[1] N is defined a constant but this value can be different at different depths and/or CU locations. As invalid encoding modes for a specific CU can be ignored, this will not be indicated in the constant N for readability.

In a typical encoder, only the coding mode with the lowest Lagrange cost will actually be used for encoding. To this end, denote $I_{m,n}$ the optimal RD cost for a CU at index m after evaluating n modes, i.e.:

$$I_{m,n} = \min(J_{m,1}, ..., J_{m,n}). \tag{4.5}$$

The computational complexity required for evaluating mode n for a given CU at index m will be denoted $C_{m,n}$. When evaluating all CU configurations and coding modes, the total frame complexity is given by $\sum_{m=1}^{\Omega} \sum_{n=1}^{N} C_{m,n}$.

4.3.1 Defining and simplifying the optimization problem

One way to reduce complexity is to not evaluate all encoding modes / depths. To this end, denote α_m the actual number of encoding modes evaluated for a CU at index m ($\alpha_m \in \{1, ..., N\}$). Note that the value of α_m can have an influence on the list of evaluated CUs and therefore change Ω.

Clearly, when only a limited set of encoding modes is evaluated (i.e., with $\alpha_m < N$), there is a possible penalty in terms of RD cost. This penalty will be called the RD cost error E_{m,α_m}:

$$E_{m,\alpha_m} = I_{m,\alpha_m} - I_{m,N}. \tag{4.6}$$

The RD cost error is zero when the optimal encoding mode is within the first α_m encoding modes, otherwise E_{m,α_m} will be positive. The challenge is now to intelligently define these α_m so that rate and distortion are optimized while the reduced complexity C_R does not exceed a particular complexity threshold C_T. In other words:

$$C_R = \sum_{m=1}^{\Omega} \sum_{n=1}^{\alpha_m} C_{m,n} \tag{4.7}$$

$$\min_{\alpha_0,...,\alpha_\Omega} \sum_{m=1}^{\Omega} E_{m,\alpha_m} \text{ subject to: } C_R \leq C_T. \tag{4.8}$$

Solving this constrained optimization problem is complex, particularly since the RD cost (and therefore also I_{m,α_m} and E_{m,α_m}) in general depends on decisions taken in the context of previously coded CU's. If the number of encoding modes to evaluate has been severely restricted for a particular CU, the RD performance for that CU could be relatively low, which increases the RD cost for spatially and temporally neighboring CU's trying to exploit correlation with this CU.

In this chapter, α_m is calculated as follows. First, we simplify the problem by ignoring the dependency discussed in the previous paragraph, and assuming that

the value for α_m can be decided independently from other CU's. In other words, we assume that the α_m's are independent from each other as well as the resulting E_{m,α_m}'s. In addition, the goal of our technique is to distribute the Lagrange cost error E_{m,α_m} equally across the frame. To cope with different CU depths, we normalize the Lagrange cost error by dividing it through the number of pixels at a certain depth. As such, denote $\bar{E}_T$ the normalized Lagrange cost error target that we want to achieve, per pixel. Also, denote $\bar{E}_{m,\alpha_m}$ the normalized Lagrange cost error per pixel.

This way, we reformulate Eq. 4.8 as:

$$\min_{\alpha_0,\ldots,\alpha_\Omega} \sum_{m=1}^{\Omega} |\bar{E}_T - \bar{E}_{m,\alpha_m}| \text{ subject to: } C_R \leq C_T. \tag{4.9}$$

Based on this equation we build a complexity controller that will vary the target Lagrange error cost $\bar{E}_T$ frame-by-frame in an attempt to match the complexity target C_T. Then, given $\bar{E}_T$ at the start of coding a frame, we estimate $\bar{E}_{m,\alpha_m}$ for each CU and determine when to stop encoding.

4.3.2 Complexity Controller

In the previous section we assumed that there are N encoding modes. From this section onwards, this is simplified to two groups of encoding modes for explaining the concept of the complexity controller and the experimental results. However, this does not mean that this technique cannot work with more encoding modes. The two groups that are used are respectively the 2Nx2N merge mode and another group containing all other encoding modes.

In 2Nx2N merge mode, a specific MV can be selected from a merge candidate list. Next to the MVs, the merge candidate list also contains a link to the reference picture list and reference picture index. This allows more precision on top of similar modes in H.264/AVC. The skip flag in 2Nx2N merge mode indicates that the residual information will not be encoded in the stream. Therefore, the minimal amount of symbols for encoding a CU is encoding the merge flag, merge index and the split flag. The 2Nx2N merge mode has relative low complexity and is used frequently.

4.3.3 Deciding when to stop encoding

The encoder estimates $\bar{E}_{m,\alpha_m}$ to decide whether or not to continue evaluating other modes, where $\bar{E}_{m,\alpha_m}$ is a normalized version of Eq. 4.6. While $I_{m,n}$ is available at mode n, $I_{m,N}$ is not available and therefore needs to be estimated. Figure 4.5 illustrates the relationship between $I_{m,n}$ and $I_{m,N}$ for depth 0 after 2Nx2N merge mode. Due to efficient construction of the merge candidate list,

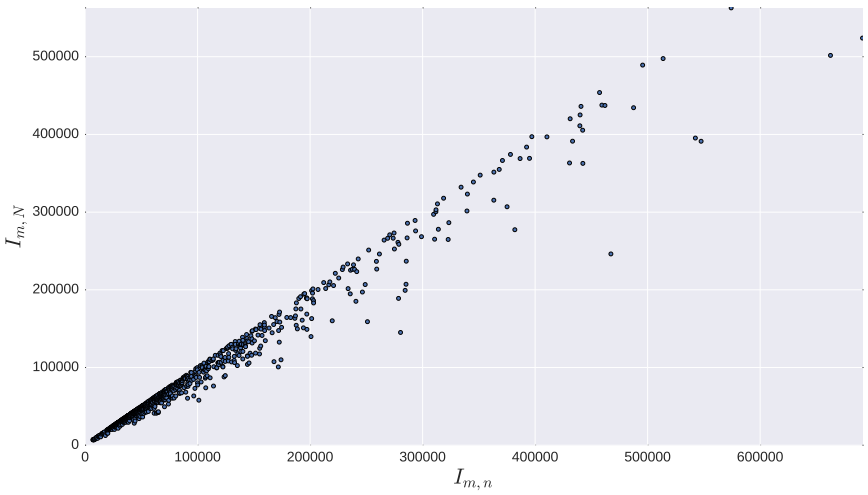


Figure 4.5: Scatter plot of measurements of $I_{m,n}$ and $I_{m,N}$ after 2Nx2N merge mode for CUs at depth 0.

there is a high probability that 2Nx2N merge mode is the optimal encoding mode or has low RD cost difference with the optimal mode. So, high correlation is expected. Therefore, in this chapter a simple linear model is used to predict $I_{m,N}$. The coefficients of the linear model for depth $\{0, 1, 2, 3\}$ are respectively $\{0.87, 0.92, 0.94, 0.95\}$. A higher coefficient indicates that on average a lower $\bar{E}_{m,\alpha_m}$ is expected if further evaluation of encoding modes is not done. In future work, it is expected that this model can be extended with higher precision by using more features of the CUs.

4.3.4 Designing the complexity controller

The goal of the complexity controller is to respect the complexity target C_T. There is a relation between C_R and $\bar{E}_T$. This relation is illustrated in Figure 4.6. An anchor encoder is used to extract I_{m,α_m} and the encoding is re-run with multiple thresholds $\bar{E}_T$. When $\bar{E}_T = 0$ (i.e. no RD cost error in the frame), the complexity is reduced only on the CUs where the 2Nx2N merge mode was the optimal mode. When $\bar{E}_T > 0$, different CUs will stop encoding after mode α_m when I_{m,α_m} is below the normalized threshold ($\bar{E}_T$). The complexity reaches a minimum when all CUs (at depth 0) only evaluate 2Nx2N merge mode, further increasing the threshold will not decrease the complexity anymore. With a prediction model, $\bar{E}_T$ can be defined for a constrained complexity C_T. But, this model is not available for the complexity controller in a real-time implementation. However, C_R and $\bar{E}_T$ are respectively an increasing and a decreasing monotonic function of α_m (see

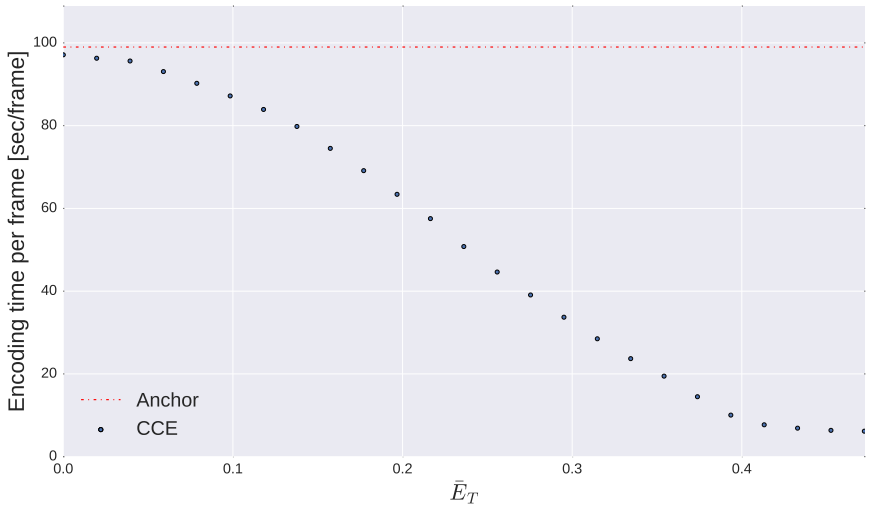


Figure 4.6: Reduced encoding time per frame for the kimono sequence and QP=12

Eq.4.8). So, Increasing the number of evaluated encoding modes α_m will always increase C_R and decrease $\bar{E}_T$. Therefore, the complexity controller can use $\bar{E}_T$ to control the frame complexity C_R, due do the properties of monotonic functions.

This is a typical root finding problem. The root in this case is the point $\bar{E}_T$ where the difference between C_R and C_T is minimal. However, there are additional difficulties compared to conventional root finding algorithms. In this use case, while $\bar{E}_T$ is constant for each frame, it typically changes slightly for every frame in the same sequence due to the changing motion complexity of the video or inaccurate measurements of the encoding complexity. Therefore, we need a complexity controller that adapts $\bar{E}_T$ continuously on a frame-by-frame basis. In a conventional root-finding problem the algorithm finds a point closer to the root at each iteration. In this case, only a single iteration can be used each frame and the algorithm can fine-tune this value on the next frame. Therefore, the basic complexity controller proposed in this chapter is based on the root-finding bisection method. This proposed method is explained with pseudo-code in Figure 4.7. If the root is outside the current interval (i.e. the last two measurements), σ will be doubled if the previous iteration was in the same direction (i.e. σ has the same sign). Doubling σ allows the method to faster search for the correct interval. When the interval with the root is found, σ is divided by -2 (i.e. changing the direction), allowing more precise approximation of the correct value while still being able to adapt to local changes in the video or noise on the complexity measurement. Finally, $\bar{E}_T$ is incremented with the current σ value. As $\bar{E}_T = 0$ indicates maxi-

$\sigma \leftarrow \sigma_0$
$equal\_sign \leftarrow True$
function UPDATETHRESHOLD(C_T, C_R)
  if $(\sigma \geq 0) \neq (C_R > C_T)$ **then**
    $\sigma \leftarrow \sigma / -2$
    $equal\_sign \leftarrow False$
  else
    if $equal\_sign$ **then**
      $\sigma \leftarrow 2\sigma$
    end if
    $equal\_sign \leftarrow True$
  end if
  $E_T \leftarrow \max(0, E_T + \sigma)$
end function

Figure 4.7: Pseudo code of the proposed complexity controller

mum complexity, it is not useful to accept negative values for $\bar{E}_T$. Note that σ and $equal\_sign$ can be initialized with any value (e.g. respectively σ_0 and $True$), this will only have influence on the iterations needed to find the correct E_T value.

4.4 Experimental results

The proposed method is evaluated using sequences from class B described in the common test conditions of HEVC [82] (i.e. Kimono, ParkScene, Cactus, BQTerrace, BasketballDrive). The HM 16.2 reference encoder is used as the anchor. We focus on low delay settings (i.e. main tier and P slices only), with QP values 22, 27, 32, 37 and open GOP structure.

Figure 4.8 and Figure 4.9 shows the encoding time per frame for QP=22 for respectively the anchor method and the proposed method. For the anchor, the measurements show a lot of variation between the different sequences. The Cactus sequence (highest encoding time) uses 42.8% more encoding time as the Kimono sequence (lowest encoding time). There is also variation within a sequence. The frame with the highest complexity in the BasketballDrive sequence has 14.1% more encoding time compared to the frame with the lowest complexity. For the proposed method (Figure 4.9), a complexity target C_T of 40 seconds per frame was defined. As can be observed in the figure, the encoder initializes during the first frames in an attempt to find the RD cost threshold using the bisection method, as described in Section 4.3.2. After this initialization phase, the complexity controller adapts the threshold frame-by-frame, and manages to keep the computational complexity close to the complexity budget, with an average deviation of 0.57 seconds (1.4%). Reduced complexity comes at a limited cost (see Table 4.1), e.g., 0.13 dB

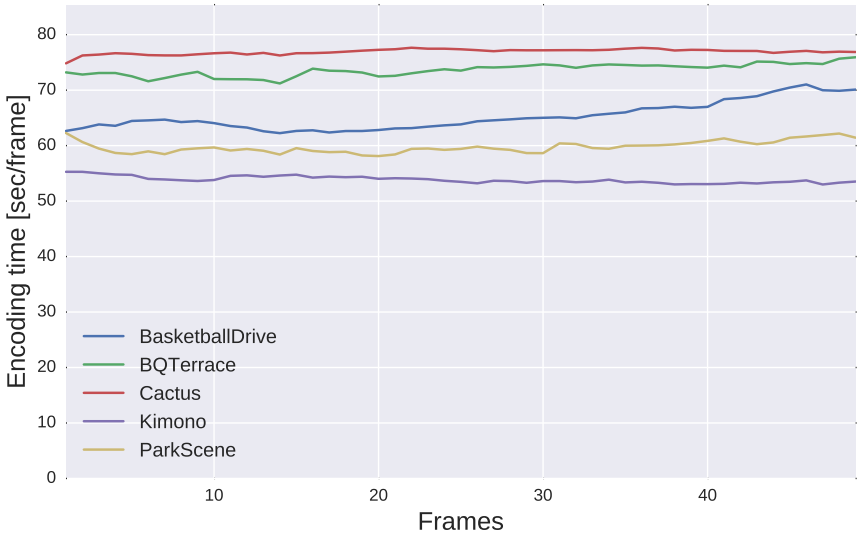


Figure 4.8: Encoding time per frame for class B sequences using the anchor method. Up to 41% difference in encoding time is measured between test sequences using the same encoder configuration.

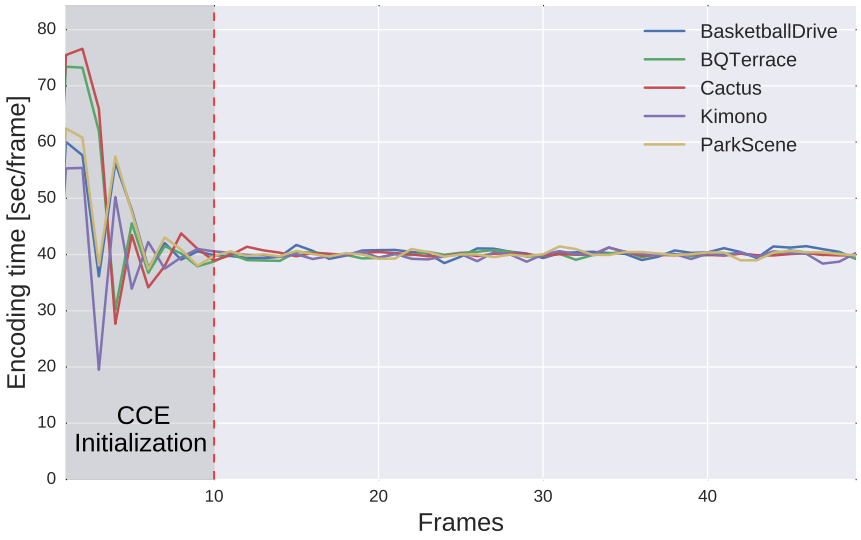


Figure 4.9: Encoding time per frame for class B sequences using the proposed method. After the initialization phase, the encoder time per frame is approximately 40 sec for all sequences.

PSNR loss and 1.6% bitrate loss for a complexity reduction of 48% for the Cactus sequence.

	Anchor			CCE		
Sequence	ET (sec)	PSNR (dB)	Rate (Mbit/s)	ET (sec)	PSNR (dB)	Rate (Mbit/s)
basketballdrive	65.56	42.69	55.04	40.26	42.59	53.13
bqterrace	73.89	42.93	135.55	39.95	42.62	137.52
cactus	77.09	41.70	92.99	40.10	41.57	94.52
kimono	53.71	43.63	12.69	39.87	43.59	12.49
parkscene	59.82	42.26	20.99	40.09	42.18	20.88

Table 4.1: Comparison of anchor method and proposed method with a complexity target of 40 sec.

Figure 4.10 and Figure 4.11 show RD curves for different complexity targets (C_T), for respectively the kimono and parkscene sequence. The C_T is defined relative to the average anchor encoding time at same QP point (anchor encoding time for each QP are provided in the Figure). For a complexity target of 80%, the BD-rate increase ranges from 0.00% (for ParkScene) to 1.72% (for BqTerrace).

The proposed method for complexity control is compared with a number of fixed complexity methods incorporated in the HM encoder (see Table 4.2). Using early skip detection (ESD) [63], an average complexity of 74.8% (relative to the anchor) is measured. When only a maximum CU depth of 2 (MD2) is allowed, an average encoding time of 41.8% of the anchor encoding is measured. At high complexity targets, lower BD-rates are measured compared to the ESD method. The BD-rate loss at $CT = 40\%$ is lower for the BasketBallDrive, Cactus and ParkScene sequences. While our technique provides complexity control, and in general better RD performance compared to the ESD and MD2 methods. Occasionally, the latter methods show better performance, indicating that there is still room for improving our technique further. This is part of future work, in which improvements could include, for example, improving the RD cost error prediction and/or allowing more groups of encoding modes instead of only 2Nx2N merge mode.

The proposed method has been ported to the x265 encoder [24] to demonstrate real-time encoding on constrained systems. When only 2 threads are used on an Intel Xeon CPU E5-2620 v3 running at 2.40GHz (no C-states enabled), the x265 encoder is only running real-time when configured on a bitrate smaller than 500 kbps. With the proposed method, x265 can encode real-time at all bitrates. The PSNR loss is illustrated for the kimono sequence in Fig. 4.12. For a bitrate of 2309 kbps, there is 0.75 dB PSNR loss with 44% complexity reduction.

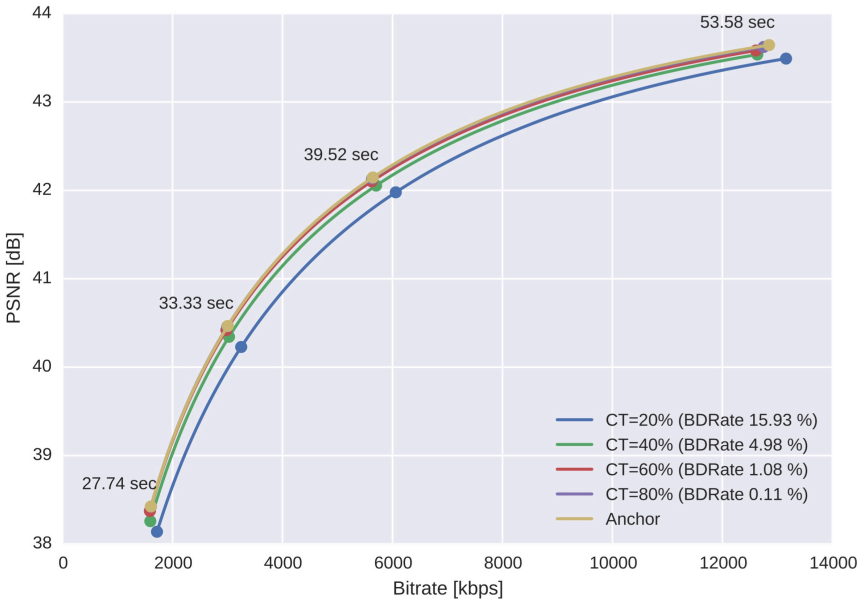


Figure 4.10: RD-curve comparison when different complexity targets (C_T) are used for the kimono sequence.

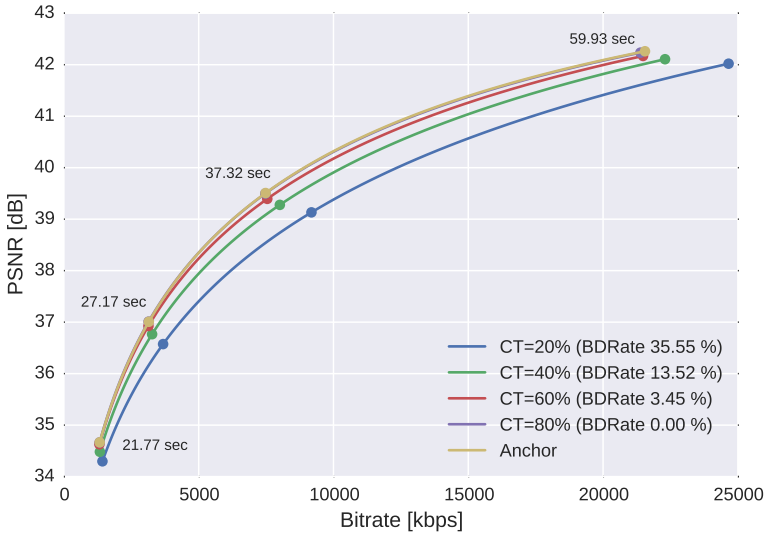


Figure 4.11: RD-curve comparison when different complexity targets (C_T) are used for the parkscene sequence.

Sequence	Proposed Method								Fixed Complexity Reduction			
	80%		60%		40%		20%		ESD [63]		MD2	
	ET	BDR	ET	BDR	ET	BDR	ET	BDR	ET	BDR	ET	BDR
basketballdrive	0.79	0.07	0.59	1.34	0.40	6.47	0.20	28.23	0.74	2.22	0.42	8.98
bqterrace	0.79	1.72	0.59	5.48	0.39	11.17	0.20	24.33	0.77	1.56	0.43	6.90
cactus	0.78	0.18	0.59	2.83	0.39	10.49	0.20	28.92	0.76	2.00	0.41	12.20
kimono	0.79	0.11	0.59	1.08	0.40	4.98	0.20	15.93	0.78	2.37	0.40	3.24
parkscene	0.79	0.00	0.59	3.45	0.40	13.52	0.20	35.55	0.69	2.28	0.43	14.18

Table 4.2: BD-Rate comparison between the proposed method and build-in methods for fixed complexity reduction (Early Skip Detection (ESD) and Maximum CU depth of 2(MD2))

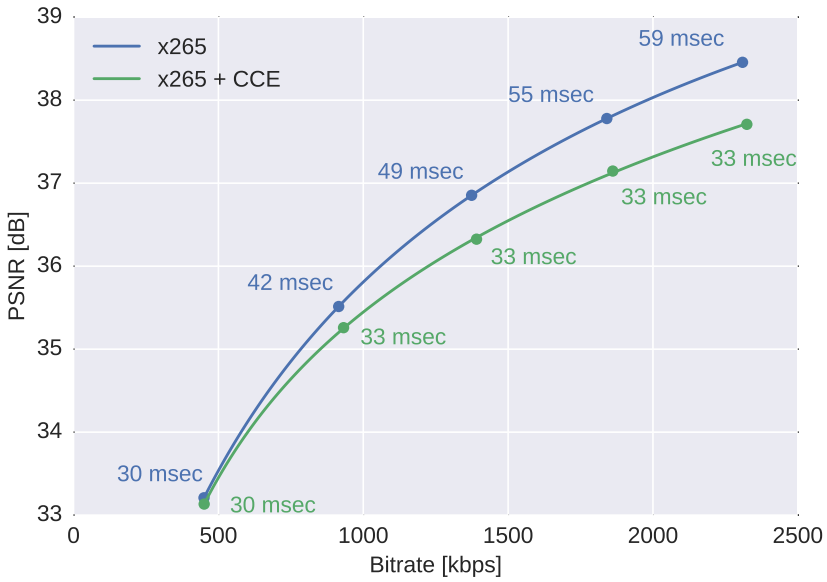


Figure 4.12: RD curves with average encoding time per frame for Kimono sequence with x265.

4.5 Conclusions and original contributions

In this chapter we proposed a technique to control the computational complexity of an HEVC encoder. This technique consisted of adaptively deciding between 2Nx2N merge mode or full evaluation, based on a dynamic threshold on the expected RD cost error. Our method operates independently of the host system and the video sequence, and enables constraining the encoding complexity to a given complexity target (after a short initialization phase). Therefore, this technique allows to use software encoding in complexity constrained environments with optimal usage of the available computational complexity (e.g. cloud encoding). The proposed technique is evaluated in two H.265/HEVC implementations (e.g. HM and x265) and the results show that the technique is effective, providing fast convergence while incurring only a limited loss in RD performance.

Possible extensions on this method have already been identified and are planned as future work. As an example, while currently the complexity controller can for a given CU only stop evaluating more encoding modes after 2Nx2N merge mode it could be extended to stop after other encoding modes. Another improvement that could be made is using a better prediction method for the RD cost error. Currently only the CU depth and RD cost of 2Nx2N merge mode is used to predict the RD cost error, this could be extended to also use other features already calculated by the encoder (e.g. motion vector information for lower CU depths, neighbouring

CU features, information from collocated CUs in previous frames). Lastly, there is also room for improvement in the complexity controller. By introducing sub-frame complexity control, the complexity controller could better adapt to sudden changes in the video sequence (e.g. scene changes).

The work on complexity constrained encoding has led to the following publications:

- T. Vermeir, J. Slowack, G. Van Wallendael, P. Lambert and R. Van de Walle, *Low Delay Complexity Constrained Encoding*, 2014 IEEE Data Compression Conference (DCC), Utah, 2016.
- T. Vermeir, J. Slowack, G. Van Wallendael, P. Lambert and R. Van de Walle, "Real-time complexity constrained encoding," 2016 IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, 2016, pp. 819-823.

The following patents were filed:

- T. Vermeir, UK 1605130.2, *Complexity control*, filing date: 25/03/2016

5 Overall Conclusions

5.1 Overall Conclusions

In the past, video solutions for professional applications have been deployed using dedicated networks and hardware due to the high requirements towards latency, quality, and reliability. In recent years, the high performance of consumer-level CPUs, GPUs, etc. make it more and more interesting to integrate off-the-shelf components into professional video solutions. However, successfully integrating consumer electronics demands some additional research to satisfy the requirements of computationally demanding applications such as professional video processing.

Hardware acceleration for video encoding / decoding in consumer electronics typically only supports chroma subsampled video content. However, these chroma subsampling artifacts are disturbing in some professional applications dealing with screen content, so one of the challenges in this disseration was to come up with a technique to reduce these chroma artifacts. Different methods were proposed in Chapter 3. A first method was based on extending the guided image filter by calculating the filter coefficients adaptively. These filter coefficients were then embedded in the coded stream (e.g., as SEI messages in H.265/HEVC). This method already resulted in important visual improvements for screen content, at the cost of additional bit rate. The second approach, labeled guided chroma reconstruction, was essentially a decoder-side spatial upscaling filter that did not require any additional information from the encoder. As a result, this method can also be used

to enhance screen content delivered by third party encoders (e.g., on consumer electronics). Like the guided image filter, this technique exploits the correlation between the luma and chroma components, taking into account the chroma subsampling filter the encoder has used. Guided chroma reconstruction only adapts a pixel value if there is a high probability that particular correlation assumptions are correct. This second method provides improvements for screen content coded using lossless and lossy video coding schemes. For camera captured content, the loss is negligible, so there is no need for pre-classifying video content (i.e., natural vs. screen content) and the filter can be always on.

Recently, there were a lot of technological innovations that have an effect on the importance of the chroma subsampling problem. One observation is that display technology has continuously improved, increasing pixel resolution and density. As a result, chroma subsampling artifacts have essentially become smaller and therefore less visible. Another evolution that has somewhat decreased interest in chroma reconstruction is that for some use cases (e.g. SCADA) old systems are replaced with newer systems that provide dedicated apps for remote control, these apps do not use video streaming and therefore do not have problems with chroma subsampling. In addition, from a more technical point of view, a lot of newer computer generated content does not have the typical properties of screen content as described in Chapter 3. This is even the case for modern desktop systems where the windowing system uses for example transparency, antialiasing and shadows on window borders. As this type of content has characteristics that are more similar to camera captured content, the proposed method does not have high gains in chroma reconstruction. Despite the descreased importance of chroma reconstruction in some fields, chroma reconstruction and high image quality in general remains a key requirement in markets like Healthcare. Also, in retail and advertisement, synthetic content is often displayed on large LED walls (e.g. the digital billboards on Times Square in New York). Depending on the pixel density of these walls, as well as the viewing distance, chroma subsampling artifacts can be visible. For these reasons, within Barco, development is currently ongoing to design a real-time FPGA implementation for chroma reconstruction. This development is based on the guided chroma reconstruction technique proposed in Chapter 3.

When hardware-acceleration on consumer electronics is not available (e.g., absent or already in use by a different process), video coding can be performed purely in software. However, the computational complexity of video encoding in software is typically very high and variable, due to changing characteristics in the video sequence. Therefore, a second challenge in this dissertation was to study techniques for controlling computational complexity in an H.265/HEVC software encoder. The proposed technique consisted of adaptively deciding between 2Nx2N merge mode or full evaluation of all encoding modes, based on a dynamic threshold on the expected RD cost error. The method operates independently of the host sys-

tem and the video sequence, and enables constraining the encoding complexity to a given complexity target (after a short initialization phase). Therefore, this technique allows to use software encoding in complexity constrained environments with optimal usage of the available computational complexity. The results show that the technique is effective, providing fast convergence while incurring only a limited loss in RD performance.

In Barco, several demos of complexity constrained encoding have been set up on both high-end and low-end laptops. The problem, stuttering video (as a result of a complexity bottleneck), is very noticeable and therefore the solution has been well received even by a non-technical audience. Quality degradation per frame is considered less disturbing compared to variable framerate, especially for high motion content. Due to these demos, some new use cases were identified where complexity constrained encoding can be used. For example, when video processing services need to be deployed flexibly in a cloud environment, the exact hardware (and hence available encoding resources) may not be known in advance. By using complexity constrained encoding, full frame rate is guaranteed regardless of the platform. Another advantage for complexity constrained encoding is that it can provide lower latency compared to some hardware accelerated platforms. Typically, hardware acceleration implementations have five to ten frames of latency. The solution of complexity constrained encoding looks promising for ClickShare, Barco's wireless presentation system, and efforts are ongoing to integrate complexity constrained encoding in ClickShare.

The techniques in this research have been demonstrated using H.265/HEVC Main profile, but the ideas are in fact more general and can be applied to other video coding standards as well. H.265/HEVC was selected because its predecessor, H.264/AVC, was one of the major general purpose video coding standards supported on all consumer electronics when this PhD project (2012) started. While H.265/HEVC is now supported on a lot of consumer electronics, there are more bumps on the road than expected. There are still issues with patent rights, as the price to use H.265/HEVC as a replacement for H.264/AVC is significantly higher. As a counteraction, a group of companies formed the Alliance for Open Media to develop royalty free video codecs. At the time of writing, it is still uncertain in which direction the future of video coding will evolve.

The research in this PhD project started with the question how we can integrate consumer electronics in professional video systems in a cost effective way. Now at the end of this dissertation, this challenge is even more prominent and more challenging. Video capabilities of consumer electronics keep improving at a high pace. Online video providers are now the first to push the boundaries of the video content they deliver to consumer electronics. For example, these platforms are pioneering with: higher resolutions, stereoscopic video, 360° video, augmented and virtual reality, etc. Moreover, these service providers offer new functionality

at a very inexpensive cost or even free (as in free beer). The capabilities of the consumer electronics (smart-phones, tablets, laptops, smart-TVs) are now greatly outperforming the classical broadcast market. As a consequence, consumers will demand even more from professional video systems in the future and this emphasizes the need for cost effective integration of consumer electronics in professional video systems.

5.2 Publications

Following publications were made during this PhD project:
Publications in international journals

- T. Vermeir, J. Slowack, R. Van Belle, S. Van Leuven, G. Van Wallendael, J. De Cock, R. Van de Walle, *Guided Chroma Reconstruction for Screen Content Coding*, in IEEE Transactions on Circuits and Systems for Video Technology. Accepted for future publication.
- J. De Praeter, G. Van Wallendael, T. Vermeir, J. Slowack, P. Lambert, *Spatially misaligned HEVC transcoding with computational-complexity scalability*, in Journal of Visual Communication and Image Representation, Oct 2016, pp. 129-158.

Publications in international conferences

- T. Vermeir, J. Slowack, S. Van Leuven, G. Van Wallendael, J. De Cock and R. Van de Walle, *Adaptive guided image filtering for screen content coding*, 2014 IEEE International Conference on Image Processing (ICIP), Paris, 2014, pp. 5561-5565.
- T. Vermeir, J. Slowack, G. Van Wallendael, P. Lambert and R. Van de Walle, *Low Delay Complexity Constrained Encoding*, 2014 IEEE Data Compression Conference (DCC), Utah, 2016.
- T. Vermeir, J. Slowack, G. Van Wallendael, P. Lambert and R. Van de Walle, "Real-time complexity constrained encoding," 2016 IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, 2016, pp. 819-823.

Submissions to MPEG/JCT-VC standardization

- T. Vermeir, *Use cases and requirements for lossless and screen content coding*, JCTVC-M0172, Incheon, Korea, April 2013.
- T. Vermeir, J. Slowack, J. De Cock, G. Van Wallendael, S. Van Leuven, *AhG8: Guided Image Filtering for Screen Content Coding*, JCTVC-N0148, Vienna, Austria, July 2013.

Patents

- T. Vermeir, US 20150016720, *Guided image filtering for image content*, publication date: 15/01/2015.
- T. Vermeir, US 14/107257, *Efficient error recovery*, publication date: 18/06/2015.
- T. Vermeir, WO 2016113396, *Method and apparatus for chroma reconstruction*, publication date: 21/07/2016.
- T. Vermeir, UK 1605130.2, *Complexity control*, filing date: 25/03/2016.

Bibliography

[1] *Netflix*. https://www.netflix.com/browse, 2016. [Online; accessed 10-August-2016].

[2] *YouTube*. https://www.youtube.com, 2016. [Online; accessed 10-August-2016].

[3] *Vimeo: Watch, upload and share HD and 4k videos with no ads*. https://vimeo.com, 2016. [Online; accessed 10-August-2016].

[4] *Display and visualization solutions: Displays, Projectors, Video walls, LED displays, Monitors & Custom imaging — Barco*. https://www.barco.com/en/, 2016. [Online; accessed 10-August-2016].

[5] *Digital cinema projectors*. https://www.barco.com/en/Products/Projectors/Digital-cinema-projectors, 2016. [Online; accessed 10-August-2016].

[6] *Star Trek Beyond*. http://www.ready2escape.com, 2016. [Online; accessed 10-August-2016].

[7] Barco. *Auro11-1*. https://www.barco.com/en/Auro11-1, 2016. [Online; accessed 10-August-2016].

[8] http://www.barco.com/en/clickshare, 2016. [Online; accessed 10-August-2016].

[9] *QAWeb — Barco*. https://www.barco.com/en/QAWeb, 2016. [Online; accessed 22-September-2016].

[10] Barco. *Uncompressed OR-over-IP platform*. https://www.barco.com/en/Products/Networked-solutions/Nexxis-for-the-operating-room/Uncompressed-OR-over-IP-platform.aspx?, 2016. [Online; accessed 10-August-2016].

[11] A. Kumcu, L. Vermeulen, S. A. Elprama, P. Duysburgh, L. Platisa, Y. Van Nieuwenhove, N. Van De Winkel, A. Jacobs, J. Van Looy, and W. Philips. *Effect of video lag on laparoscopic surgery: correlation between performance and usability at low latencies*. International Journal Of Medical Robotics and Computer Assisted Surgery, page 10, 2016.

[12] C. Poynton. *Digital Video and HDTV Algorithms and Interfaces*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1 edition, 2003.

[13] William B. Pennebaker and Joan L. Mitchell. *JPEG Still Image Data Compression Standard*. Kluwer Academic Publishers, Norwell, MA, USA, 1st edition, 1992.

[14] ISO/IEC. *ISO/IEC 11172-1:1993 - Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – Part 1: Systems*. 1993.

[15] ITU-T. *H.261 : Video codec for audiovisual services at p x 384 kbit/s*. Nov 1988.

[16] ITU-T. *H.262 : Information technology - Generic coding of moving pictures and associated audio information: Video*. Technical report, Feb 2012.

[17] ITU-T. *ITU-T SG16: Multimedia*. http://www.itu.int/en/ITU-T/studygroups/2013-2016/16/Pages/default.aspx, 2016.

[18] L. Chiariglione. *MPEG, The Moving Picture Experts Group website*. http://mpeg.chiariglione.org, 2016.

[19] W. Li. *Overview of Streaming Video Profile Amendment in MPEG-4 video standard*. In MPEG-4. 2001 Proceedings of Workshop and Exhibition on, pages 75–78, 2001.

[20] DVB Project. *Standards - Digital Video Broadcast*. https://www.dvb.org/standards, 2016.

[21] Alliance for Open Media. *Alliance for Open Media*. http://aomedia.org, 2016.

[22] ITU-T. *H.265: Series H: Audiovisual and multimedia systems: Infrastructure of audiovisual services - Coding of moving video: High efficiency video coding*. April 2015.

[23] JCT-VC. *HEVC Test Model (HM) HM-16.3*. https://hevc.hhi.fraunhofer.de/HM-doc/, 2016. [Online, accesed 3-August 2016].

[24] MulticoreWare Inc. *x265*. http://x265.org/, 2016. [Online, accessed 3-August 2016].

[25] *MulticoreWare — Accelerated Software & Development Services*. http://www.multicorewareinc.com, 2016. [Online; accessed 2-October-2016].

[26] *VideoLAN - x264, the best H.264/AVC encoder*. http://www.videolan.org/developers/x264.html, 2016. [Online; accessed 2-October-2016].

[27] MulticoreWare Inc. *MulticoreWare demonstrates high quality 4K 10-bit real-time HEVC video encoding with x265*. http://www.design-reuse.com/news/37109/multicoreware-4k-10-bit-real-time-hevc-video-encoding-with-x265.html, 2015. [Online, accessed 3-August 2016].

[28] Struktur AG. *Github - strukturag - libde265*. https://github.com/strukturag/libde265, 2016. [Online, accesed 3-August 2016].

[29] *GNU Lesser General Public License, version 3*. `https://www.gnu.org/licenses/lgpl-3.0.en.html`, June 2007. [Online; accessed 2-October-2016].

[30] GStreamer Developers. *GStreamer: open source multimedia framework*. https://gstreamer.freedesktop.org, 2016. [Online, accessed 3-August 2016].

[31] VideoLAN non-profit organization. *VideoLAN - Official page for VLC media player, the Open Source video framework!* http://www.videolan.org/vlc/index.html, 2016. [Online, accessed 3-August-2016].

[32] Microsoft. *DirectShow (Windows)*. https://msdn.microsoft.com/en-us/library/windows/desktop/dd375454(v=vs.85).aspx, 2016. [Online, accessed 3-August-2016].

[33] Struktur AG. *libde265.js JavaScript HEVC/H.265 bitstream decoder*. http://www.libde265.org/blog/2014/04/10/libde265-js-javascript-hevch-265-bitstream-decoder/, 2016. [Online, accesed 3-August 2016].

[34] Fabrice Bellard. *FFmpeg*. https://ffmpeg.org/, 2016. [Online, accessed 3-August-2016].

[35] *GNU General Public License, version 3*. `http://www.gnu.org/licenses/gpl.html`, June 2007. [Online; accessed 2-October-2016].

[36] IETR/INSA Rennes. *OpenHEVC/openHEVC: HEVC decoder*. https://github.com/OpenHEVC/openHEVC, 2016. [Online, accesed 3-August 2016].

[37] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. *RTP: A Transport Protocol for Real-Time Applications*. STD 64, RFC Editor, July 2003. `http://www.rfc-editor.org/rfc/rfc3550.txt`.

[38] H. Schulzrinne and S. Casner. *RTP Profile for Audio and Video Conferences with Minimal Control*. STD 65, RFC Editor, July 2003.

[39] H. Yu, X. Want, and Y. Jing. *Doc. JCTVC-M0320: More investigation on screen content*. Technical report, JCT-VC, Incheon, Korea, April 2013.

[40] Apple. *Apple - iPad Air - Technical Specifications*. https://www.apple.com/ipad-air/specs/, 2014. [Online; accessed 19-August-2014].

[41] Google. *Supported Media Formats, Android Developers*. http://developer.android.com/guide/appendix/media-formats.html, 2014. [Online; accessed 19-August 2014].

[42] D. Flynn, D. Marpe, M. Naccari, T. Nguyen, C. Rosewarne, K. Sharman, J. Sole, and J. Xu. *Overview of the Range Extensions for the HEVC Standard: Tools, Profiles, and Performance*. IEEE Transactions on Circuits and Systems for Video Technology, 26(1):4–19, Jan 2016.

[43] D. Flynn, D. Marpe, M. Naccari, T. Nguyen, C. Rosewarne, K. Sharman, J. Sole, and J. Xu. *Overview of the Range Extensions for the HEVC Standard: Tools, Profiles, and Performance*. IEEE Transactions on Circuits and Systems for Video Technology, 26(1):4–19, Jan 2016.

[44] J. Xu, R. Joshi, and R. A. Cohen. *Overview of the Emerging HEVC Screen Content Coding Extension*. IEEE Transactions on Circuits and Systems for Video Technology, 26(1):50–62, Jan 2016.

[45] T. Vermeir. *Doc. JCTVC-M0172: Use cases and requirements for lossless and screen content coding*. Technical report, JCT-VC, Incheon, Korea, April 2013.

[46] F. Walls and S. MacInnis. *VESA Display Stream Compression*, Jul 2014.

[47] Microsoft. *Remote Desktop Protocol*. https://msdn.microsoft.com/en-us/library/aa383015, 2016.

[48] T. Richardson. *The RFB Protocol, Version 3.8*. http://www.realvnc.com/docs/rfbproto.pdf, Aug 2008.

[49] Y. Wu, S. Kanumuri, Y. Zhang, S. Sadhwani, G. J. Sullivan, and H. S. Malvar. *Tunneling High-Resolution Color Content Through 4:2:0 HEVC and AVC Video Coding Systems*. In Proceedings of the 2013 Data Compression Conference, pages 3–12, Washington, DC, USA, 2013.

[50] K. Ugur, D. Bugdayci, and M. M. Hannuksela. *Doc. JCTVC-P0121: On frame packing arrangement SEI message for 4:4:4 content in 4:2:0 bitstreams*. Technical report, JCT-VC, San Jose, US, January 2014.

[51] Y. Itoh and T. Ono. *Up-sampling of YCbCr4:2:0 image exploiting inter-color correlation in RGB domain*. Consumer Electronics, IEEE Transactions on, 55(4):2204–2210, November 2009.

[52] M. Zhao, P. M. Hofman, and G. de Haan. *Content-adaptive up-scaling of chrominance using classification of luminance and chrominance data*. In Electronic Imaging 2004, pages 721–730. International Society for Optics and Photonics, 2004.

[53] C. Tomasi and R. Manduchi. *Bilateral filtering for gray and color images*. In Computer Vision, 1998. Sixth International Conference on, pages 839–846, 1998.

[54] K. He, J. Sun, and X. Tang. *Guided image filtering*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(6):1397–1409, 2013.

[55] F. Ono, W. Rucklidge, R. Arps, and C. Constantinescu. *JBIG2 - The ultimate bi-level image coding standard*. In Image Processing, 2000. Proceedings. 2000 International Conference on, volume 1, pages 140–143, 2000.

[56] D. Flynn, C. Rosewarne, and K. Sharman. *Doc. JCTVC-O1006: Common test conditions and software reference configurations for HEVC range extensions*. Technical report, JCT-VC, Geneva, Switzerland, October 2013.

[57] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. *Image quality assessment: from error visibility to structural similarity*. Image Processing, IEEE Transactions on, 13(4):600–612, April 2004.

[58] J. Ohm, G. J. Sullivan, H. Schwarz, K. T. Thiow, and T. Wiegand. *Comparison of the Coding Efficiency of Video Coding Standards; Including High Efficiency Video Coding (HEVC)*. Circuits and Systems for Video Technology, IEEE Transactions on, 22(12):1669–1684, Dec 2012.

[59] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand. *Overview of the High Efficiency Video Coding (HEVC) standard*. Circuits and Systems for Video Technology, IEEE Transactions on, 22(12):1649–1668, 2012.

[60] G. Correa, P. Assuncao, L. Agostini, and L.A. da Silva Cruz. *Performance and Computational Complexity Assessment of High-Efficiency Video Encoders*. Circuits and Systems for Video Technology, IEEE Transactions on, 22(12):1899–1909, Dec 2012.

[61] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra. *Overview of the H.264/AVC video coding standard*. Circuits and Systems for Video Technology, IEEE Transactions on, 13(7):560–576, July 2003.

[62] K. Choi and E. S. Jang. *Fast coding unit decision method based on coding tree pruning for high efficiency video coding*. Optical Engineering, 51(3):030502–1–030502–3, 2012.

[63] J. Kim, J. Yang, K. Won, and B. Jeon. *Early determination of mode decision for HEVC*. In Picture Coding Symposium (PCS), 2012, pages 449–452, May 2012.

[64] M. Kim, H.-J. Lee, and N. Ling. *Fast merge mode decision for diamond search in High Efficiency Video Coding*. In Visual Communications and Image Processing (VCIP), 2013, pages 1–6, Nov 2013.

[65] Z. Pan, S. Kwong, M.-T. Sun, and J. Lei. *Early MERGE Mode Decision Based on Motion Estimation and Hierarchical Depth Correlation for HEVC*. Broadcasting, IEEE Transactions on, 60(2):405–412, June 2014.

[66] C.S. Kannangara, I.E. Richardson, and A.J. Miller. *Computational Complexity Management of a Real-Time H.264/AVC Encoder*. Circuits and Systems for Video Technology, IEEE Transactions on, 18(9):1191–1200, Sept 2008.

[67] G. Correa, P. Assuncao, L. Agostini, and L. A. da Silva Cruz. *Complexity scalability for real-time HEVC encoders*. Journal of Real-Time Image Processing, pages 1–16, 2014.

[68] T. Zhao, Z. Wang, and S. Kwong. *Flexible Mode Selection and Complexity Allocation in High Efficiency Video Coding*. Selected Topics in Signal Processing, IEEE Journal of, 7(6):1135–1144, Dec 2013.

[69] O. Lehtoranta and T. D. Hamalainen. *Complexity analysis of spatially scalable MPEG-4 encoder*. In System-on-Chip, 2003. Proceedings. International Symposium on, pages 57–60, Nov 2003.

[70] C.-G. Zhou, I. Kabir, L. Kohn, A. Jabbi, D. Rice, and X.-P. Hu. *MPEG video decoding with the UltraSPARC visual instruction set*. In Compcon '95.'Technologies for the Information Superhighway', Digest of Papers., pages 470–477, March 1995.

[71] T. Katayama, W. Shi, T. Song, T. Shimamoto, and J. S. Leu. *Reference frame selection algorithm of HEVC encoder for low power video device*. In 2016 2nd International Conference on Intelligent Green Building and Smart Grid (IGBSG), pages 1–6, June 2016.

[72] G. He, D. Zhou, Y. Li, Z. Chen, T. Zhang, and S. Goto. *High-Throughput Power-Efficient VLSI Architecture of Fractional Motion Estimation for Ultra-HD HEVC Video Encoding*. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 23(12):3138–3142, Dec 2015.

[73] D. Zhou, L. Guo, J. Zhou, and S. Goto. *Reducing power consumption of HEVC codec with lossless reference frame recompression*. In 2014 IEEE International Conference on Image Processing (ICIP), pages 2120–2124, Oct 2014.

[74] Y. Lee, J. Kim, and C. M. Kyung. *Energy-Aware Video Encoding for Image Quality Improvement in Battery-Operated Surveillance Camera*. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 20(2):310–318, Feb 2012.

[75] H. Touzani, A. Mansouri, F. Errahimi, and A. Ahaitouf. *Implementation analysis of HEVC encoding on dual ARM Cortex A15 architecture*. In 2016 International Conference on Information Technology for Organizations Development (IT4OD), pages 1–5, March 2016.

[76] F. Saab, I. H. Elhajj, A. Kayssi, and A. Chehab. *Profiling of HEVC encoder*. Electronics Letters, 50(15):1061–1063, July 2014.

[77] A. Mativi, E. Monteiro, and S. Bampi. *Memory access profiling for HEVC encoders*. In 2016 IEEE 7th Latin American Symposium on Circuits Systems (LASCAS), pages 243–246, Feb 2016.

[78] G. Correa, P. Assuncao, L. A. da Silva Cruz, and L. Agostini. *Encoding time control system for HEVC based on Rate-Distortion-Complexity analysis*. In 2015 IEEE International Symposium on Circuits and Systems (ISCAS), pages 1114–1117, May 2015.

[79] Intel Inc. *Intel Xeon Processor E5-2620 (15M Cache, 2.00 GHz, 7.20 GT/s Intel QPI)*. http://ark.intel.com/products/64594/Intel-Xeon-Processor-E5-2620-15M-Cache-2_00-GHz-7_20-GTs-Intel-QPI, 2016. [Online; accessed 19-August-2014].

[80] Canonical Ltd. *Ubuntu 14.04.5 LTS (Trusty Tahr)*. http://releases.ubuntu.com/14.04/, 2016. [Online; accessed 19-August-2014].

[81] Intel Corporation. *Power Management States: P-States, C-States, and Package C-States*. https://software.intel.com/en-us/articles/power-management-states-p-states-c-states-and-package-c-states, 2014. [Online; accessed 19-August-2014].

[82] F. Bossen. *Doc. JCTVC-L1100: Common test conditions and software reference configurations*. Technical report, JCTVC, Geneva, Switzerland, January 2013.